UNIVERSITY OF OSLO
Faculty of Humanities
Department of Linguistics and
Scandinavian Studies
(Humanistic Informatics)

Getting back on the trail, combating information overload with Topic Maps

Thesis for the Cand. Philol. degree

Rolf B. Guescini

May 8, 2006

Contents

1 Historical overview and contributions **1**

1.1 Vannevar Bush 1

1.1.1 Memex 2

1.1.2 Sequential vs. Associative 2

1.1.3 From Memex to hypertext 3

1.2 Theodore Nelson 4

1.2.1 Literary Machines 4

1.2.2 Project XANADU 5

1.2.3 Embedded markup 6

1.2.4 Other visions and projects on the way 8

1.3 Douglas Engelbart 12

1.4 Hypertext before the World Wide Web 14

1.4.1 Modularity, juxtaposing, and editing 14

1.4.2 Hierarchical structure vs. non-hierarchical link structures . 15

1.4.3 Filtering of information 16

1.4.4 Extended link functionality 16

1.4.5 Paths 17

1.4.6 High level info to combat overload 18

1.4.7 Tim Berners-Lee and the World Wide Web 18

1.4.8 Development of the World Wide Web 19

1.4.9 WWW becomes commercial 20

1.4.10 The World Wide Web Consortium 21

2 The World Wide Web and HTML: What is wrong with hypertext at this point? **23**

2.1 The hyper in hypertext 24

2.1.1 Associative links 25

2.1.2 Link directionality 26

2.1.3 Broken links 26

2.2 To structure or not to structure 27

2.2.1 Structure vs. Presentation 27

2.2.2 Logical . . . 28
2.2.3 Presentational . . . 29
2.2.4 Distinction blurred . . . 29
2.2.5 Markup or Browser; where to place the blame? . . . 31
2.3 A parallel . . . 31
2.4 High level information and Polyscopy to combat information overload? . . . 32
2.5 If not HTML, then what? . . . 34

3 Metadata **37**
3.1 Processing information : Human vs. computers . . . 37
3.2 Why do we need metadata on the web? . . . 38
3.3 What to use metadata for . . . 39
3.4 Meta, but how to create it? . . . 42

4 New Markup Languages for a new era **45**
4.1 Standardization . . . 45
4.2 XML, the eXtensible Markup Language . . . 45
4.2.1 Checking documents for Well-Formedness . . . 46
4.2.2 Checking documents for validity . . . 46
4.3 XML, the future document format? . . . 47
4.3.1 XML for Web documents . . . 47
4.3.2 XML for other media . . . 49
4.4 XML based markups . . . 50
4.5 The new era of the Semantic Web . . . 50

5 The Resource Description Framework, and the Semantic Web **53**
5.1 RDF vs. XML . . . 57
5.2 RDF and then... . . . 57
5.3 What does RDF solve? . . . 59

6 Subject-based classification: Controlled vocabularies or Ontologies? **61**
6.1 Controlled vocabularies . . . 61
6.2 Ontologies . . . 62
6.3 What do Ontologies solve? . . . 63

7 The Web Ontology Language, W3C's language for Ontologies **65**
7.1 What does OWL solve? . . . 66

8 Topic Maps **67**
8.1 Topics . . . 68
8.2 Scoping topics . . . 69
8.2.1 Typing topics . . . 69
8.3 Associations . . . 69
8.3.1 Members playing roles . . . 70
8.4 Occurrences . . . 70
8.5 Reification . . . 71
8.6 Identity . . . 71
8.6.1 Published Subject Indicators . . . 72
8.7 Merging Topic Maps . . . 72
8.8 Topic Map notations . . . 73
8.9 What do Topic Maps solve? . . . 73

9 RDF or Topic Maps? **77**
9.1 Similarities . . . 77
9.2 Differences . . . 78
9.2.1 Why choose Topic Maps? . . . 80
9.2.2 A possible hypertext implementation? . . . 80

10 TMemex **83**
10.1 Metadata for me? . . . 83
10.2 Implementing the trail . . . 85
10.2.1 A useful addition to the trail? . . . 86
10.3 Technical Solution . . . 86
10.3.1 The thought browser . . . 86
10.3.2 Use case: populating the trail . . . 88
10.3.3 Use case: Using the trail . . . 90
10.4 Implementation of the browser . . . 91
10.4.1 The parallel document windows . . . 94
10.4.2 The Topic Maps engine . . . 94
10.4.3 Building the trail . . . 95
10.4.4 Rendering the trail and its functions . . . 97
10.5 Future possibilities . . . 100

11 Summary and conclusion **101**

Appendices **103**

A SQL schema for the Topic Map Trail **105**

B TopicMap Object Interface **109**

C Javascript / AJAX code **115**

D HTML implementation of the browser interface **131**

List of Figures

2.1 Information fragmentation 33
2.2 The Polyscopic Information Ideogram 34

5.1 Syntactic tree 55
5.2 The direction of the arc tells us what resource is the subject or the object. 56
5.3 Multiple RDF statements. 56
5.4 Layer architecture of the Semantic Web. (Koivunen and Miller, 2002) 58

8.1 Binary Topic Map Association (Garshol, 2003) 70
8.2 Reifying an association (Garshol, 2003) 71

9.1 The two standards families (Garshol, 2003) 78

10.1 The thought arrangement of windows in the browser 87
10.2 Use case of user populating a trail 89
10.3 Use case of user using an existing trail 92
10.4 The Touch Graph applet visualizing a trail 93
10.5 Screen shot of add document dialogue 96
10.6 Screen shot of highlighting of in-document selections 99

Acknowledgment

This thesis is written partly as a result of studies in Hypermedia at the Department of Linguistics and Scandinavian Studies of the University of Oslo, Norway, specifically its division for Humanistic Informatics dealing with Computer Science, Artificial Intelligence and Computer Linguistics. It created an unique environment in which also studies into hypermedia were allowed to blossom. It is these studies among other things that have inspired me, and which culminate into this thesis. I'd like to thank my advisor **Kåre A. Andersen** for his vision and good work in trying to teach me and other students what was of real importance to considerate when producing hypertext and hypermedia in order to combat information overload. His contributions in helping me sort out the directions of this thesis has been invaluable.

Part of this thesis is also very much a result of my inspiring collaboration with **Dino Karabeg** of the Department of Informatics of the University of Oslo, Norway. Karabeg has been working for a decade on developing the Information Design Methodology in an attempt to create a more structured and high level approach to information modeling. He sees the conscious creation of information of uttermost importance to the augmentation of human intellect, securing sustainability of our shared resources, and our cultures. I would like to thank Karabeg for creating an environment for the development of these thoughts in the Information Design class, and for being a true humanist, inspiration and friend. Most of all for including me in his important work, leading to parts of this thesis becoming possible.

Norway has been in the forefront of development of Topic Map technologies through the invaluable efforts made by several Norwegian companies, most notably **Ontopia**, and especially through the contributions of **Steve Pepper** and Lars Marius Garshol of Ontopia. I would like to thank Ontopia for giving me valuable insight as well as free coursing in the details of Topic Mapping. Specifically, I would like to thank **Lars Marius Garshol** for always being patient and helpful

whenever I have turned to him for his invaluable insight into the more subtle issues of Topic Mapping.

Abstract

Vannevar Bush identified the current problem of Information Overload already in the 1930's and with his 1945 article "As we may think", he proposed a solution that would bring the machine to man's aid and help the individual create her personal order out of the emerging informational chaos. The current state on the World Wide Web of today shows us that its implementation of hypertext is not an adequate solution to the problems that Bush identified. It is therefore partly the goal of this thesis to go back to Bush's inital ideas, specifically the notion of the *trail*, to see how they have inspired the pioneers of hypertext, and the evolution of hypertext before the World Wide Web. This thesis will also look into contemporary technical tools created to address the problem, as well as the author's own contribution in the development of an information design methodology believed to help guide the conscious design of information to combat information overload. The methodology sees the importance of designing structured high level information as entry points to the vast amount of dispersed and detailed information, and proposes Topic Maps as the right tool for doing it. As the thesis' practical approach, the author presents a protype browser, "TMemex", implementing Bush's Memex by using Topic Maps to see if we can obtain an ordering concept for the individual using the current resources of the World Wide Web closer to what Bush envisioned.

Introduction

The Information Age is upon us, and has been for many years already. We are drowning in information resources produced by a multitude of different media supposed to make life easier for us. Information has always been abundant, but the situation we are living in today is something exceptional seen from a historical point of view. Never before has the production of information been so prolific, as what we are experiencing today. Historically there was always a limited set of individuals or an authoritative organization of some kind that produced the information being published for public consumption. Today the possibility for anyone to produce and publish information available for anyone is much greater.

Information Overload

The birth of Hyper Text Markup Language and World Wide Web in 1990 lead to the possibility for anyone to easily produce documents and to publish them within seconds. It did of course take some years before it caught on for the ordinary man, as it was just the "web literate" that produced web pages in the beginning, as it has been with all new media throughout the history. But producing web pages has truly become the most open information channel ever created by humans. Today the global amount of indexable [1] web pages published on the Internet has reached astronomic figures. According to a recent study (Gulli and Signorini, 2005) which used web searches in 75 different languages to sample the Web determined there were over 11.5 billion web pages in the publicly indexable web, including both personal home pages, and home pages published by entities such as newspapers, companies, organizations e.t.c. With such an abundance of information available, and the possibility for anyone to produce whatever they feel like, it is inevitable that any piece of information risks being duplicated at least once, and with no versioning systems, or any control on the quality of the information we are effectively drowning in information and experiencing information overload, a term coined by

[1] The part of the web which is considered for indexing by the major search engines

Toffler (Toffler, 1970). Which generally means having more information than we can readily assimilate.

There are several explanations offered to this phenomenon, one is usually offered by cognitive psychologists; the concept of "technostress" (Wikipedia, 2006a) tells us that perceived technostress induces a correlate perception that users are being controlled by "ICT", or Information and communications Technology, rather than being empowered by it. It is said to result in the same problems as any other kind of stress; reduced intellectual performance and poor judgment. In (Conklin, 1987), Jeffrey Conklin writes that reading a hypertext tends to present the reader with a large number of choices about which links to follow and which to leave alone, and these choices lead to a certain overhead that Conklin describes as

> "Cognitive Overhead", the additional effort and concentration necessary to maintain several tasks or trails at one time.
> J. Conklin

Conklin goes on saying that this problem is not something that came into being with hypertext. We have seen before that the brain can create ideas faster than other of our attributes can process them, hypertext simply offers a more effective tool to engage the mind with the richness of creative thought, which is a might be a drawback when it is not needed or wanted.

Lost in hyperspace

Cognitive Overhead is a problem not only related to information overload, but is also related to a similar problem that evolved with the birth of hypertext, a phenomenon called "lost in hyperspace.", also described in (Conklin, 1987). Along with the power of being able to organize information with much greater complexity than before hypertext, comes the problem of knowing

(a) where you are in the network

(b) how to get to some other place that you know (or think) exists in the network

As we have limited memory, keeping up with several pieces of information at once might be a problem, what Conklin calls a “disorientation problem”. One thing is managing the mass of information contained within one page, but since hypertext gives us the ability to make arbitrary jumps from one page to another page, which has an arbitrary relation to some piece of information contained

within the originating page, new problems arise. We might be able to keep track of one, maybe two jumps out of our original context, but as soon as we jump further and further out of the originating context, our short term memories are just not powerful enough. The user often experiences disorientation and a sense of loss of context which arises from unfamiliarity with the conceptual structure and organization of the site they are taken to.

Another related phenomenon, is having browsed and done several searches, but still not being able to summarize or reproduce what one has learned, nor having any detailed memory of particular items, due to a web site's sheer vastness, referred to as the "Art Museum Phenomenon", Foss, (1998). Krug, (2000) states that given context overload, or being lost in hyperspace, users tend to attempt a navigation strategy of choosing the first link which appears to suit their requirements. This behavior has been called "satisficing" after the decision-making practices of firefighters, who have to choose the best available action under extreme pressure. This again will lead the user into a new context, adding to the confusion.

Problem not solved

In (Levy, 2005), David Levy references a 2003 report by Varian and Lyman that estimates the amount of new information stored on paper, film, magnetic and optical media to have doubled in between 1999 and 2002. Levy's findings points out that the development of digital information systems and global hypertext seems not to have solved the problem Vannevar Bush identified in his famous 1945 article "As we may think", but instead has exacerbated it. The technologies that Bush foresaw and hoped would tame the problems which were recognized already then, has maybe contributed to the intensification of the problem. Levy goes on saying that it could be argued that it isn't the sheer amount of information in the world that itself is the problem, rather it is the difficulty in gaining access to and managing what is most relevant, and that the digital tools haven't kept pace with the rate of expansion.

It seems as though HTML and our current implementations of hypertext in general isn't working to our benefit after all, it seems as though they are not the right tools in them selves to combat information overload, being the situation that Bush identified as a future problem in the 1930's and 1940's, and which still has not been solved. Bush meant that we needed associative structuring of information to improve on the accessibility of dispersed information resources, crossing interdisciplinary boundaries and what he considered as the artificiality of indexing

systems of his time. HTML and the World Wide Web as the present implementation of associative linkage just connects dispersed information, and makes it possible to make a jump between these resources, and has shown itself to be a good tool for that exact purpose. But it seems though that arbitrary associative linking the way we are doing today is a wrong model for organizing information in order to combat information overload.

The rest of this thesis will look back at Bush's initial thoughts and at different historical approaches taken to implement hypertext, and to discuss if it is hypertext in itself or our current implementation of it that is failing. I will also try to propose a thought implementation of a prototype closer to what Bush envisioned. In chapter 1, I will give a historical overview of the pioneers of hypertext and their contributions to the field, identifying hypertextual features thought to be useful which are not present in our present implementation of hypertext. Chapters 2 and 3 will try to identify what problems HTML and our current implementation of hypertext suffers from as well as make a suggestion to necessary amendments, hereby presenting the author's contribution to the field of information design in the form of the participation in papers proposing an information design methodology for combating information overload. Chapters 4 - 9 will present current tools that are thought to improve the tools for our information needs of the future. In chapter 10 I will discuss the implementation of an imagined browser, a modern Memex, "TMemex", based on Topic Maps to help build Bush's notion of the trail. A tool to create personal metadata in the advent of more fundamental changes to happen on the World Wide Web.

Chapter 1

Historical overview and contributions

1.1 Vannevar Bush

More than 60 years ago, Vannevar Bush wrote the "As we may think" article, where he spoke about the ever increasing amount of information that our society generates. Already in the early 1930s, when he began working on the idea of Memex, the amount of information gathered was staggering and still growing, and Bush foresaw that we in the future would get severe problems both remembering all the information we take in, but also finding relevant information in the "infoglut"[1]. He foresaw that we would need some kind of system with which we could store that information efficiently, but with which we also would be able to navigate it efficiently.

In (Bush, 1945), he says that the problem is not so much the fact that we are producing or even over-producing information, the problem lies in that we are not able to absorb the information, or to access it efficiently. Since Bush was a researcher himself, one task that was important to him was having the possibility to share information swiftly and efficiently with other researchers worldwide. Bush wrote that much of human development had been slowed down, even lost, because one person's findings never would reach the intended public, in order for them to carry on the idea and develop it further. Either the information would not reach them at all, or too late making the findings outdated or useless.

[1]Word used in (Pepper, 2000). Glut means according to WordNet "the quality of being so over-abundant that prices fall", Its meaning together with information would mean something like "information being so abundant that the quality of it is diminishing"

1.1.1 Memex

So Bush wished to extend the power of human beings by creating radically new ways of communicating and working together, and saw that much of the research and technical advances done during the second world war could be put to the benefit of civilians worldwide. This depended on the possibility of sharing the information with other researchers in other parts of the world, in order for them to be able to do further development. He saw that technology could bring us

> "a new relationship between thinking man and the sum of our knowledge"

one that would promote

> "the application of science to the needs and desires of man"

Bush saw our limited memory as one of the main problems that had to be addressed, and envisioned a machine that would help us in our task of ordering, storing and accessing our increasing amount of information. Bush thought that a machine of this sort, would help us save important information and ideas which otherwise would be lost for us. He envisioned it as a sort of desktop machine which could use microfilms, pictures and sound, and providing a way of saving this information for later retrieval. He envisioned recording all literature available, such as dictionaries, atlases etc. onto microfilm, the new and promising medium of his time. Seeing the technical advances already happening in his own time, he thought that a machine that would work in many ways as the human mind does could be built in the future as an aid to our feeble memories.

1.1.2 Sequential vs. Associative

Not only did Bush see that we were going to drown in information, but he also saw that the traditional structuring of information was not suited to human thought. Bush meant that the human brain was not built to absorb information in the traditional sequential way it was presented in traditional literature, but that the human mind works rather by association. A now famous extract of his article goes like this:

> "...The human mind does not work that way. It operates by association. With one item in its grasp, it snaps instantly to the next that is suggested by the association of thoughts, in accordance with some intricate web of trails carried by the cells of the brain...."

So he envisioned the Memex as a device that would extend our natural minds and help us in perceiving information closer to how the human brain actually works, by arbitrary association. The device was thought to have two screens which could present information gathered from the Memex, and by pointing on the screens, two items of information were to be permanently tied together and to have a connection or what we today know as a link between each other upon retrieval. This was to be the essential feature of the Memex, something he named "associative indexing", an analogy to the way the human mind snaps instantly from one associated item to another.

To prevent valuable "trains of thought" being lost due to our feeble memories, he envisioned the concept of trails. These trails were to be virtual "trains of thought" made up by several items connected together by association, recorded on top of the information,and to be recallable at any time to represent the "train of thought" which otherwise would have gotten lost. The trail was of course envisioned to be possible to record on a microfilm, making it possible for reuse in another person's Memex, where the trail might be viewed as is, or even made to become a side trail of one of the owner's trails.

1.1.3 From Memex to hypertext

Bush's article has since its publication been an inspiration to other pioneers on the way towards the hypertext we are living with today. People like Ted Nelson, Douglas Engelbart and Tim Berners-Lee, all having an important impact on the way our hypertext systems function today, were all inspired by some or other level of Bush's ideas, and have done their part in bringing Bush's vision towards its goal. Ted Nelson can be said to have picked up on Bush's idea that information should be structured closer to how our brains work, Douglas Engelbart has been instrumental in creating the tools that we use in the future Bush foresaw. Tim Berners-Lee has in implementing a working hypertext standard and the protocols to support it, seen the importance of a framework to allow for collaboration amongst peers to support a rapid sharing of important research and information.

There are questions to be asked and answered though, have we really reached a point where our tools actually are improving on the information overload we are experiencing. Was Bush right in his idea that information would be more accessible if we organized information by his notion of associative indexing? And if he was, have we managed to implement his ideas in a way that is sufficient grounds for saying whether he was right or wrong? One cannot say that the present World Wide Web and its applications are direct implementations of the Memex

neither in them selves nor seen together. One could say that parts of Bush's vision has been implemented through the present hypertext systems, but seeing to what extent they are solving our information overload problems, it is interesting to see if it is Bush's ideas themselves that aren't a sufficient solution to the problems, or if it is our implementations of his ideas that are done badly, and thus are adding to the problems instead of improving on them.

1.2 Theodore Nelson

Theodore Nelson, inspired by Bush's thoughts on associative information structures, invented the term "hypertext" in 1965 and is a pioneer of information technology and within the field of research on hypertext. He founded Project Xanadu in the early 1960's, and has written among others, two books; "Computer Lib / Dream Machines" (1974) and the 1991 "Literary Machines" that documents and discusses his view on hypertext and the Xanadu system. Nelson has also since the birth of the WWW, HTML and what he defines as embedded markup, expressed his dislikes with it, regarding it as gross over-simplification of his own work.

> "HTML is precisely what we were trying to PREVENT – ever-breaking links, links going outward only, quotes you can't follow to their origins, no version management, no rights management."
> (Nelson, 2005)

1.2.1 Literary Machines

When Nelson coined the word hypertext in 1965 as "nonsequential writing", it was a result of many years of thought, drawing lines from observations of several fields. He didn't see hypertext as confined only to digital documents residing on computers, but rather the more general thought of non-sequential writing. To Nelson, the front page of a newspaper or magazine layout with text and inset illustrations could be considered a hypertext. He finds writing sequential literature difficult because there are too many possible connections to be done within the text, and thus to decide the correct sequence of the different parts of the text. Also, he finds that reading sequential non-fictional texts points out to us that our thoughts work non-sequentially, since the active reader often "skips ahead, jumps around, and ponders about background material". (Nelson, 1993a)

So he saw hypertext as the solution to the unnaturalness of the examples above to the human mind, in that one does not have to decide on sequential structure

when presenting information, but rather on inter-connective structure which provides much greater flexibility. Then throughout the years of doing his education and trying to get his ideas through to the world, he would constantly work on and try to develop his sense of hypertext, resulting in several stages of thought, culminating into the idea for his Xanadu project.

1.2.2 Project XANADU

Nelson thought that Bush's notion of trails was too bound by his interest in the use of microfilms for the Memex, leading to his notion of the trail having a sequence.(Nelson, 1972) Nelson thought that with the new digital storage, no sequence needed to be imposed on the trail, and instead of storing materials in their order of arrival or of being noticed, it ought to be possible to create overall structures of a greater useful complexity. This being the essence of what Nelson thinks of as hypertexts, namely, *"non-sequential writing"*. The imposition of sequence or otherwise other organizing scheme on information is something Nelson frowns upon, thinking that it rather than empowering the user when trying to make sense out of information, it limits her, making it difficult to perceive content the way the mind works naturally.

In (Nelson, 1972) Nelson says that Bush's "As we may think" article has been generally misinterpreted, and that what Bush wrote and thought has little to do with what we call information retrieval as prosecuted today (1972). He goes on saying that Bush did not think well of indexing and that he instead discussed new forms of interwoven documents, and finds it strange how Bush's article has been taken so to heart in the field of information retrieval since it, according to his interpretation runs counter to virtually all work being pursued under the name of information retrieval. This is of course Nelson's interpretation, as he himself rejects any form of organizing scheme or indexing on information as wholly unnatural. When Bush said that "... our ineptitude in getting at the record is largely caused by the artificiality of systems of indexing...", it could also be interpreted as him thinking that the current implementations of indexing systems were artificial, and not necessarily that indexing was artificial altogether

Nelson's own idea of how information ought to be organized and presented was to be implemented in his own vision, namely the XANADU hypertext project. It was all based on non-sequentiality and proposed a whole system for how documents could be saved and interlinked. Where no sequence was to be imposed on the material making it possible to create new structures by linking copies of existing documents in the system together to form new documents. Nelson proposed

a complex and rich system of linkage where typing and multi-directionality was intended to help create many types of presentations or documents on basis of the same pieces of information.

The Xanadu Parallel Textface was to be a stand-alone system sitting on the personal computer. A simple console to handle notes, writing, correspondence, reading and the creation of new kinds of text. It was to permit various types of screen animation, automatic retrieval and data-base editing, which was supposed to service different user front ends, or as Nelson put it "faces or theaters". The foremost of these theaters, was the Parallel Textface, a text system having many of the features described in Engelbart's article of 1962 (Engelbart, 1962)

The user was to sit at a display screen with a typewriter keyboard, a light pen or other pointing tool, and other various controls to be used for reading, exploring, annotating, writing, and revising. Storage was to be digital, where the system was to be able to manipulate the words letter by letter, rather than as a single image which was what was the standard at his time of writing. It may seem like nothing much in light of the technology of today, but it was pretty much futuristic at the time Nelson envisioned it. Also, he thought of having a versioning system where the user's edits on her documents are recorded automatically in a cumulative editorial log. As Bush spoke of having several documents in different screens or panels of the same screen for simultaneous viewing of documents, the same was to hold for the Parallel Textface with the possibility for explicit linkages between associated texts. It was important that the user had the possibility to create links between text sections regardless of whether or not they were part of the same text unit or otherwise related. He also talks of the possibility of attaching type codecs, annotations, or even versioning to the links, giving the author the possibility to give the user various options of jumping and branching, reflecting any pattern of reading the author wants to make available to the user. The only constraints put on the author, aught to be usefulness, clarity and artfulness. (Nelson, 1972)

1.2.3 Embedded markup

In (Nelson, 2001), Nelson discusses one of the things he thinks less of on the web, which is Embedded markup. He says that if the advocates of SGML wish to enforce a universal, linear representation of hierarchical structure, this is an intolerable imposition which drastically curtails the representation of non-hierarchical data structures. As an example of non-hierarchical structures, he is thinking of his idea of transpublishing or transclusion presented below in 1.2.4, where documents can be built by including virtual pieces of information written by other authors.

Nelson thinks that embedded markup interferes with transclusive re-use, because among other things, an arbitrary section of HTML-marked up data may not have correct tags when taken out of context and embedded within one's own markup. Also the problem of including structure from another document into one's own document, be it of any structure, or not, causes problems. As an alternative, he proposes something he calls "parallel markup" for sequentially formatted objects, where the data has no tags embedded. Instead the markup should be in a parallel stream or medium containing reference positions in the text data stream. This is an approach that he believes has several advantages, because the data itself is left uncluttered, making it possible to process it in several different ways, not being bounded by the structure of the data. A noteworthy parallel that shows how this idea might be sound, is the division between markup and layout information that was enforced by the problem posed by cluttered up HTML code infesting the WWW. Since we are living with the WWW, where most documents are already marked up, he sees an alternative approach that leads the same way, where one would just ignore the structural elements while parsing the document, and then process in the same way as described above.

Nelson believes that embedded structures enforces sequences and hierarchy, limiting the kinds of structure that can be expressed. He asks if there is a *real* structure to things or documents, and if forcing hierarchical structures upon data will damage the original function or intention of the data. Nelson's main concern is and has always been the exact representation of human thought, and his objective is thus to create editorial systems for exact representation, where information can be formatted in a continuing, evolutionary way, Not being forced into the traditional sequential structures we have used until now. This means finding a representation of structure which recognizes anarchic and overlapping relations and which maintains structure and constancy across successive versions of the data. Nelson thinks that embedded markup like HTML cannot represent this idea at all and merely adds obstacles to solving these tasks. Nelson instead proposes a references model which breaks information apart in parallel, by handling contents, structure, and special effects or layout separately. This way the different parts can be more easily understood and worked on, and he also thinks that this way more general structures might be represented. In conclusion to these thoughts, he proposes a three-layer model reflecting his proposition for reference model, and the usage of HTML merely as an output format.

There are several things that might be learned from Nelson's insight, and which can be used as a basis for future informational applications for the web. Most notably his thoughts on linkage, but also the parallel reference model is very much

a possible way to go when structuring information. We have already seen XML's and XHTML's focus on division between logical and presentational information as solutions to the problems posed by the mixing of structural and layout information in web documents, Nelson also adds the dimension of separating content from structure in order for us to have full flexibility in working with, and presenting information on the web. Also, the point of modularization and granularity of content by making smaller pieces of information into self sufficient modules, would make it possible to attach richer semantical information to them. This is well worth keeping in mind when reading the discussion of how linking into vast documents causes information overload related problems in 2.1.1

1.2.4 Other visions and projects on the way

If we want to do research on ways of handling and designing information alternative to tradition, but also to the ways it is presently done on the WWW, Ted Nelson has done much interesting thinking on that field. Being very much suspicious to hierarchical organization, has driven him to think of information structuring in completely non-traditional ways. Reading about his projects in hindsight within the context of looking for alternative information structuring principles, one sees that many of Nelson's projects poses real alternative possibilities to the design of information. Many of these ideas were recorded on his way towards his vision of the Xanadu system, and might have been overlooked by people making our present systems, thinking that Nelson's ideas were not worthwhile contemplating since the Xanadu system itself never was completed. Seeing how the resulting systems are not solving the initial agenda set by Bush more than 60 years ago, it is worth taking a new look at Nelson's ideas.

Zippered lists Trying to combine the ideas of screen editing and idea management where you could compare alternative versions side by side on the screen and the notion of non-sequential writing, this system came to his mind at his period at Vassar. It was to be a data structure with several sequences linked together sideways permitting certain intercomparisons and certain forms of non-sequential writing. This would be almost the same as what he called "chunk style hypertext", where one had many separate paragraphs each with many branching choices. This system would allow for intercomparisons between versions, in which an item could be an important heading in one version and a trivial point in another, but seen together, the linked sequences would form a whole document, making it possible to retain the other versions at the same time. Again, Nelson shows how modularization of information would help us in creating more flexible information structures.

Links

> " A link is simply a connection between parts of text or other material. It is put in by a human. Links are made by individuals as pathways for the readers exploration; thus they are parts of the actual document, part of the writing"(Nelson, 1993b)

Nelson sees the link as actually something more than just the attachment of odds and ends, they are intrinsic to the document, and is also what enables us to create non-sequential writing or hypertext. He therefore thinks that a hypertext system should allow the user to create links of any kind, between any things the user might want to link.

Link types Nelson has a notion of typed links for his Xanadu system, where the simplest type of link would be the one where the user creates "book-marks", places where she may want to re-enter within a text when returning to it. The browser of today of course has the ability of bookmarking entire documents, but still not the ability for the user to actually create a link within the document as part of her personal trajectory through the information. Other types of links he envisions are footnote-links, marginal notes, hypertext jumps, as well as the notion of links with multiple endpoints, attached to different kinds of objects on every side of the link, or even links attached to other links. Nelson's advanced concept of linking is something that sadly would require for much of the existing infrastructure of the web to be changed, but it still is a valuable pointer to the importance of having typed links, and link types that have different functions. Especially interesting is his notion of the "book-mark" link which could, if information was modularized be used to target portions of documents.

Link Rot Nelson also thought of the problem of link rot quite early, as links are very hard to keep constantly updated with conventional computer storage structure. Nelson proposes a quite genius scheme using an idea he has for storage of data he calls "prismatic storage" or "evolutionary storage". To make a lengthy story short, he envisions versioning of content while producing documents by saving each change as as a chronological fragment, instead of saving the whole document as a block every time. Now, if one supposes Nelson's idea of a universal docuverse where a document is saved only one time, and all new usage of the same content happens through transclusion, the prismatic storage would implement the versioning part of the scheme. Lastly, if one attaches a link, not to a positional address in a given version of some content, but to specific characters or elements, the link will stay with these elements, whatever change was done to them.

The prerequisites of a completely new way of storing data makes implementation of this idea quite difficult at this point, which is sad, because it is certainly a good idea for combating the well-known "404 Document not found"-problem, frequently encountered on the WWW. But for more localized content managing systems, say in an intranet, this idea might be a very good solution to maintain linkages between chronological changes in documents.

Link Directionality Nelson thought that link directionality, if any, should be given in the link-type definition. He has a division between "out-links", which should be contained within the document, and be under control of its author, as well as "in-links", which would be under control of the author pointing to ones document. Nelson envisions the possibility to ask for a given document or a place within a document and ask "what connects here from other documents". In the case where a document has very many "in-links", it should be possible to filter them out based on e.g link type, time of linkage, author, subject etc.

The lack of explicit link directionality is probably one of the shortcomings of the World Wide Web which causes much of our present problems concerning information overload, and the aforementioned "lost in hyperspace" phenomenon. Link directionality together with the typing of links would make it possible to create semantically rich links, reducing the arbitrariness of following associations as experienced on the web of today. Many schemes have been tried to add some sense of directionality, telling us whether a link actually stays within the current context or if it leaves it, all with variable success. Still there is reason to believe that when one explicitly labels or types links, it would help on the sensation of loss of context often caused by hyperlink jumps. Also, if links were let's say bidirectional, the user would at least know more about the information items which link to each other, hopefully resulting in a lesser loss of context. Nelson's early idea of semantically richer hyperlinks are one of things which could have been considered as important in the beginning when designing our current hypertext system, but there are also disadvantages to directionality as argued in 2.1.2

HES - Hypertext Editing System The HES system was a system of dual purpose done together with Andries van Dam at Brown University in 1967. It was meant to produce printed documents nicely and efficiently, improving on the batch card editing technology of the time, but mainly it was meant to explore Nelson's ideas on hypertext. It was a pioneering hypertext system that organized data into two main types: links and branching text. The branching text could automatically be arranged into menus and a point within a given area could also have an assigned name, called a label, and be accessed later by that name from

the screen. Nelson's idea for the hypertext had a kind of hierarchical structure and also cross-references. In his key note at the "Hypertext '87 Workshop', van Dam recalls the hypertext schema showing the first signs of the "Lost in hyperspace" problem, and they were already getting the notion that the richer the hypertext, the greater the navigational problem. (van Dam, 1988). The editing part had functions for insert, delete, move, and copy, functions for branching text.

Nelson himself saw it as failed, since it had an emphasis on paper printout and formatting, and had little to do with hypertext. The system still looking to the familiar and tradition of paper output, did in Nelson's view set back the progress towards the "real on-line future", especially since the HES system turned out to be very influential since it was effectively the first visual computer text facility that beginners could use. It is interesting to see how we at a very early point had experiences with the "Lost in hyperspace"-problem, and how it was not taken sufficiently into account when designing our present hypertext systems.

Transclusion Nelson sees an anarchic but self-organizing system based on his conception of royalties and sub-royalties. Royalties are automatically monitored by the host computer network. Including various costs such as membership in the system, rental of terminal and hookup, logged-in time, per-usage costs of various facilities such as disk and memory usage, but most of all royalties payed to copyright holders of a given document. Every document in the system has an owner, and every owner is paid "a whiff of royalty" whenever somebody calls their document from the memory and displays it in words, sounds, or images. This feature was later known as the concept of Transclusion, where one rather than copying and embedding somebody others' data into your own document and thereby storing the data in two places, include a virtual copy of the original information fragment. Transclusion allows it to be stored only once, and viewed in different contexts. (Wikipedia, 2005c). The royalty was actually thought to be on every byte transmitted, and paid automatically to the owner of the fragment of information every time it was summoned. Since the copyright holder gets an automatic royalty anything might be quoted without further permission.(Nelson, 1990).

This system would lead to the production of compound documents, mostly being put together by fragments of data. These fragments would of course have to be written atomically, so that contents of one section does not interfere with the contents of another section. Meaning that linguistic measures such as anaphora[2]

[2]Linguistic measure where an instance of an expression is referring to another expression preceding it.

and cataphora[3], or references such as links outside of the fragment would not be possible. Since they would create confusion when included in a context within which the references would make no sense.

The framing problem Ted Nelson saw that creating small hypertext would be trivial, but as soon as hypertext gets bigger, being able to isolate sub-collections would be important. He saw that one needed the ability to restrict our concerns to subsections, and to be able to turn off the rest of the subsections that was outside of the focus at the moment of reading. This is what he calls the “framing problem”, being able to frame only a part of a large complex. He sees typed links as one possible solution to this problem, where the user may filter out, or reduce the context of what is shown. Another solution to this problem could also be his notion of *Stretchtext*, described in (Nelson, 1972) as “continuously variable text which never leaves the screen, but changes by small increments on user demand, growing longer and more detailed by a few words at a time.”

Again Nelson points out the importance of modularity and the need to be able to identify smaller units of information in order to give the user a natural and efficient way of consuming and creating information. This is similar to what is discussed in 2.4. These are indeed valuable points to think of when considering alternatives to how we could implement hypertext as a possible solution to the problems identified with hypertext on the World Wide Web in the following sections.

1.3 Douglas Engelbart

He thought about how the world was growing ever more complex and remembered his experience reading Bush. He began to "envision people sitting in front of displays, “flying around” in an information space where they could formulate and organize their ideas with incredible speed and flexibility." In 1963, Englebart set up his own research lab. He called it the Augmentation Research Center. Throughout the 1960s and 1970s his lab developed an elaborate hypermedia groupware system called NLS (oNLine System), later Augment. NLS facilitated the creation of digital libraries and storage and retrieval of electronic documents using hypertext. This was the first successful implementation of hypertext. Augmenting the intellectual abilities of humans through the use of technology has been important to Engelbart, and having read Bush’s “As we may think”, he saw that Bush’s notion of the trail could help people collaborate intellectually, and

[3]Linguistic measure that occurs when an expression corefers with a latter expression in the discourse.

thereby cause a change that would cause humans to become intellectually more effective. In (Engelbart, 1962), he looks back at the article and describes how he envisions some of the technical prerequisites for the Memex could be solved.

He goes on comparing Bush's idea of the Memex to a system he has been working on for himself consisting of IBM notecards, using notch coding to create index pointers on them. What is interesting, is how he identifies the need for small units of information, or what he calls "little kernels" of data, which could have meta data attached to them as well. They are organized in what he calls "restricted subject sets" which he finds are useful for classifying his thoughts. He also mentions the usefulness of creating substructures within these overall subject based structures as he modifies or expands his concepts. He also describes how it is easy to make searches for information in this structure. After having stated the usefulness of subject centered organization of information, as well as the need for having some kind of structuring applied to the data, he recognizes the need for making associative trails trough the material when wanting to record a train of thought developing while reading series of note cards. He finds that just having ordered information is not enough, there is still the need to create connection between elements found within different structures, which are arbitrary seen from the structural context of the informational elements themselves.

Several years later after the NLS / Augment project has been implemented, and the World Wide Web was born, he encourages the development of hypermedia design principles. In (Engelbart, 1995) he encourages the development of an "open hyperdocument system (OHS)", his idea for the WWW's continued evolution as he puts it. He sees all kinds of digital resources and documents to be inherently hyperdocument objects, and envisions a universal tool system using this universal knowledge base replacing all tools made especially to manage different resource types. All these objects should be possible to incorporate into presentational documents or hyperdocuments, much like in Nelson's Xanadu. He argues for these objects within a hyperdocument to have an explicit structure in which structural and logical substructures may be addressed and managed. He argues for every object, from full, aggregate hyperdocuments to the smallest units such as characters to be uniquely and unambiguously addressable to make them referenceable in any hyperdocument system. He also argues for meta level addressing put on links themselves, making links addressable subjects as well. Another interesting property that he calls for, is the possibility of filtering on content making it possible to represent the user with a flexible choice of viewing options, or being used as basis for new sequences or groupings of informational objects including objects residing in other documents.

Many of these ideas proposed by Engelbart, are identified not only by Bush and Nelson, but also by other hypertext systems made in the early ages of hypertext as we shall see in 1.4. Most of these ideas were not implemented in the current World Wide Web, but could maybe have been instrumental in aiding us in managing our informational needs and in combating information overload. As we shall see in the following section, the WWW became what it became due to the possibilities of its time and due to the goals of its creator and other forces involved in the building of what became the initial WWW. It might be that we could have profited greatly from having implemented some of the functions described by the early hypertext pioneers. Today, we have the luxury of hindsight, and are able to identify the main problems we are struggling with, as well as being able to compare the now maturing Web with older systems incorporating many of the features described by the pioneers. Also, we are in the favorable position of having new technologies and tools with which we can try to implement some of these functions to work with the already existing Web which we shall see an example of in chapter 10

1.4 Hypertext before the World Wide Web

HTML and the World Wide Web were not the first implementations of hypertext, in (Conklin, 1987), Jeffrey Conklin does a survey into the history of hypertext, looking into different implementations of hypertext based systems. From manual hypertexts found in different things like note cards, the Bible and the Talmud, dictionaries, encyclopedias and Aristotle's writings to the new digital hypertext systems born with the gradual access to more and more powerful computer hardware. All of them written or created before hypertext existed on the World Wide Web. When looking at why our current implementation of hypertext in HTML on the WWW doesn't solve the problems general hypertext was thought by the likes of Bush and Nelson to solve, it is fruitful to look at some of the essential ideas and features implemented in pre-WWW hypertext systems. I will therefore list and discuss some of the findings that Conklin did in his 1987 survey and group them by functionalities identified by many of the pioneers as useful.

1.4.1 Modularity, juxtaposing, and editing

Before Berners-Lee thought of using Internet's underlying distributed structure as the underlying model for hypertext, a hypertext system was thought to consist of these central aspects:

- A database, containing nodes and link pointers between nodes

- Windows on the computer screen, corresponding to nodes in the database on a one to one basis

- Links in the display pointing to the nodes in the database which would open the corresponding new window containing the destination node. Closing the window caused changes to the nodes content to be saved in the database.

As we can see, the initial idea was for nodes to be treated modularly and to live in their own windows, giving the possibility to juxtapose information modules. In later systems, such as Carnegie-Mellon University's ZOG and Knowledge Systems' KMS, where nodes would be viewed in a single frame at a time, the drawback was that users would become more easily disoriented since there was no spatial event corresponding to moving from frame to frame. Disorientation was greatly reduced though, if the user could move very quickly among frames and thus become reoriented with little effort, such as using back button of present web browsers. Also, editing was seen as a basic trait of hypertext systems. These are traits seen in most hypertext systems from Bush's vision of the Memex to the early browser implementations that Berners-Lee did at the early stages of his development while working at CERN.

1.4.2 Hierarchical structure vs. non-hierarchical link structures

When looking at Engelbart's "NLS", we see that files in NLS were structured in a hierarchy of segments, as well as having the possibility of establishing any number of reference links between these segments within and between files, mirroring what Engelbart did with his note cards in (Engelbart, 1962). Another system that had this ability was Randall Trigg's "Textnet" having two basic types of nodes, nodes with textual content and nodes which hierarchically organized other nodes, implementing both hierarchical trees and non-hierarchical graphs. "Textnet" also had three means of hypertext "perusal" as he puts it in (Trigg, 1991): *vertical*, following hierarchical structures, *horizontal*, following paths and links to side paths, as well as *jumping*, using an index of keywords.

These systems highlight the notion that hierarchical browsing and associative structures alone might not be effective means of navigating information, but that rather creating systems where these are put to work together might be a much better approach.

1.4.3 Filtering of information

NLS provided a feature for "viewing filters" for the file structure, making it possible to choose what depth of hierarchy one wished to display as well as truncating the number of items displayed at any time, giving the possibility of suppressing details at various levels specified by the user.

In 1980, Ira Goldstein and Danny Bobrow proposed a "Personal Information Environment" hypertext system to help software designers with the various views or "perspectives" that they could have on the evolving system. Nodes in the network was to have multiple perspectives, and also being organized into layers having "contexts" used to represent alternative designs.

The Intermedia project at Brown University proposed a construct called webs, to implement context dependent link display where every link belongs to one or more contexts, and is only visible when one of those contexts is active.

The notion that a piece of information might be perceived differently depending on what context one perceives it from, or even what view of reality one supports is important to be aware of in our present multicultural and fragmented culture. There are no constant truths that are valid for any situation or context, and we need to be able to mark up information with aspectual clues in order to create holistic information that reflect different needs. Having aspects marked up we have the possibility to filter out those aspects or contexts that are not pertinent to our needs when perceiving information at a given point, thus possibly diminishing information overload by filtering out unnecessary information, resulting in smaller, more manageable information modules.

1.4.4 Extended link functionality

Randall Trigg's "Textnet" is described as a system supporting "nonlinear text", in which documents are organized as primitive pieces of text connected with typed links to form a network similar in many ways to a semantic network. Trigg proposes a specific taxonomy of over 80 link types for use within his system, arguing that

> "the disadvantage posed by a limited set of link types was outweighed by the possibility of specialized processing on the hyperdocument afforded by a definite and fixed set of primitives."

Both Englebart's NLS and Brown University's FRESS systems would support typed or keyworded links as well as having support for bidirectional links. The later Intermedia system from Brown's was concerned with providing the user with ways of managing the increased complexity of the hypertext environment. The creators of the Intermedia system proposed that a system with multiple links emanating from the same point in a document may confuse the reader and that it might be better to have a single link icon within the material that could be quickly queried with the mouse to show the specific outgoing links, their names, and their destination nodes.

Typed links were seen as important early on as seen already in Nelson's writings. They are important to be able to create semantically richer associations diminishing the possibility of becoming "lost in hyperspace". But also as Trigg argues, if one in addition has a limited set of link types, it is easier to create a consistent system promoting recognition and familiarity which again might help reduce the "lost in hyperspace" phenomenon.

1.4.5 Paths

Randall Trigg's "Textnet" also supported the definition of paths as ordered lists of nodes to browse linear concatenations of text. The reader would be provided with default paths through the network which she could read in the suggested order, being relieved from having to make so many choices at each link. In a system called "Hyperties" developed at the University of Maryland containing interconnected articles, the system would keep track of the users' path through its network of nodes, allowing easy return from exploratory side paths. This type of path is resemblant of a mix between Bush's notion of the path and "breadcrumbs" paths, discussed in (Nielsen, 1999), probably being a valuable navigational tool which diminishes the danger of getting "lost in hyperspace" through providing contextual clues.

Bush's notion of the path has led to several different ideas being implemented in different systems, most notably to the idea of associative linking itself. There is though, a distinction between these paths either being sequential following Bushs's initial idea or completely arbitrary as an effect of associative linking. Trigg argues in (Trigg, 1991) that giving the user a default linear starting path avoids the need for the user to make too many choices, having only to make a choice when reaching an undesirable branch. This might be instrumental in diminishing what Conklin called "cognitive overhead" mentioned in the introduction. On the other hand a sequential path might be confining removing the users

freedom of association as argued by Nelson in (Nelson, 1972) as well as reducing the users overview of the context in which she is navigating. Whether to choose a sequential or a networked path will probably depend on the task at hand, as a sequential path would be useful when creating documentations or building an argument, and networked paths would be more appropriate when one wants to create the possibility of navigating the path freely.

1.4.6 High level info to combat overload

In addition to having a path system, the "Hyperties" system would also allow the user to be presented with a high level, short description of the systems articles as an intermediate position between bringing up the full article and trying to guess from the link name precisely what the article is about. This is very much the same way as it is done in newspapers to incite readers to go on reading some article. The Intermedia project at Brown's also studied what was needed to browse very large networks containing maybe thousands of nodes. They proposed two kinds of displays; a global map, which shows the entire network and allows navigation within it, and a local map, which represents a view centered on a single document and displaying its links and nearest neighbors in the web. In addition there are several levels of detail at which nodes and links can be displayed.

These projects are valuable examples of giving a reader or user a high level overview which may help in diminishing information overload. The "lost in hyperspace" and "cognitive overload" problems are as much caused by being given too many details as being caused by the nature of arbitrary associations, as I will argue in 2.4

1.4.7 Tim Berners-Lee and the World Wide Web

Berners-Lee's original vision of the WWW was of a sea of interactive shared knowledge, in which computers are Memexes whose knowledge base exists in cyberspace rather than Bush's microfilm. He had a vision of the "great brain" (Simpson, 1996) as a living organism, a sense of a dynamic, interactive information continuum that is the net and its users. Berners-Lee originally designed it as an interactive means for collaboration and augmentation, but it has instead become a static medium for hypertextual publication. Berners-Lee wanted an environment that would bring friends and colleagues closer, in that by working on the shared knowledge together one can come to better understandings. Also of importance was that it was to be universal, the fact that a hypertext link can point to anything, be it personal, local or global, be it draft or highly polished. There

was a second part of the dream, too, dependent on the Web being so generally used that it became a realistic mirror of the ways in which we work and play and socialize.(Berners-Lee, 1998). "One had to be able to jump," he later wrote, "from software documentation to a list of people to a phone book to an organizational chart to whatever.

1.4.8 Development of the World Wide Web

While working at CERN, the research center for particle physics, situated in Geneva near the French-Swiss border, Berners-Lee started working on the first version of a program that was going to be a predecessor to later development, which he called Enquire, short for "Enquire Within Upon Everything", a Victorian-era encyclopedia he remembered from childhood. He used it to keep track of programmers and what programs they were writing while developing software for CERN. He could type in pages of information, representing a person or a program or whatever, each represented by a node. Very much like the index-card system Douglas Engelbart describes in "Augmenting the human intellect". The only way to create a new node, was making a link from an existing node, much like Bush's paths. It had typed links, making it possible to describe the nature of associations. It also operated with both internal and external links, where the internal links were two-way links. Still it was just a little program sitting on one isolated computer.

The next step on the development came when he saw that to make a documentation system that could be used by different people, using different computers and different operating systems, he would have to create a system where people could write documents that did not have to follow some proprietary scheme based on which type of computer they had, or what operating system they were using, and that would have as common rules as possible for anyone using any system, at the same time as it was decentralized. Decentralization was the only way it could scale properly no matter how many people were using it, much like Internet is working through distributed packaging. Most similar hypertext systems of the time would as he saw it, be bogged down by a central database of some sort that everything had to pass through, so Berners-Lee saw that the already existing Internet would be the right medium to use for his vision. To have this happen, he and other pioneers which he managed to attract to his quest, wrote the HTTP [4] protocol and server software, needed to implement what is now known as the World Wide Web. He also saw the need of an addressing system where every resource had its own unique identifier which would make it possible to link to any

[4]Hyper Text Transfer Protocol

document or resource residing on any computer being part of Internet. Thus the URI [5] scheme was born, making this possible.

In 1990, Berners-Lee wrote the first web browser - or browser-editor rather, called "WorldWideWeb". At that time, it was living on Berners-Lee's computer, slowly spreading to other computers at CERN while he tried to talk the people working at CERN to adopt it as the best solution to their need for a documentation system that could be used centrally at CERN. Being also an editor, it was meant to implement Berners-Lee's vision of a collaborative environment, which was as important to his vision as being able to have people access and read information placed in it. As time went, and Berners-Lee tried to inspire people to create new and better browser-editor-applications for the new medium, the editor part of the browsers would regrettably be left out as the programmers would focus on doing the reader which was a much easier to implement and also what had the best potential for payback. In (Berners-Lee and Fischetti, 2000) Berners-Lee guesses that part of the reason for the editor being left out in browser implementations, was that collaboration required much more of a social change in how people worked. The result being that Bush's and Engelbart's vision for rapid collaboration, and Berners-Lee's hope for the web becoming an intimate collaborative medium would not become an integral part of how we would use the web in the coming years.

1.4.9 WWW becomes commercial

Initially, the WWW had a very slow startup, as it was mainly the hypertext community that picked up the early tools made available by Berners-Lee on Internet newsgroups, but as Berners-Lee's efforts to spread the word succeeded, interest for the web grew, and the commercial possibilities of the web came apparent to people, making the interest for developing browsers for the new promising medium much more interesting. Berners-Lee tells in (Berners-Lee and Fischetti, 2000) how he was surprised with what he calls "the near universal disdain for creating an editor". It seemed more important to add fancy display features into the browsers, such as multimedia, different colors and fonts, which would create a bigger buzz amongst users, than to create the collaborative applications that Berners-Lee envisioned and hoped for. An early example of that was NCSA's Mosaic browser, which was the first browser to be commercially known in the media, and thus also becoming what the average user would associate when thinking of the web. Some browser / editors were made, but as the main attention was put on the more entertaining and eye-catching aspects of information on the web, and

[5] Uniform Resource Identifier

since the first commercial browsers laid the ground for what the web would be perceived as amongst average users, the much needed attention to the collaborative possibilities of the new medium was lost.

1.4.10 The World Wide Web Consortium

One important thing for Berners-Lee was that the web was to be a universal resource, available to everyone. For that to happen, it was important that all the software making it possible for users to communicate freely with each other was based on universally consistent and uniform technology. If too many conflicting interests were to be able to freely form the destiny of the web, fragmentation would occur, resulting in the web ceasing to exist as a universal medium. Seeing that the web was growing and giving birth to possible conflicts caused by commercial interests trying to create their own proprietary solutions for the web, Berners-Lee saw that there would be a need for some kind of body to oversee the web's development and the development of standards to combat fragmentation, ensuring the universality needed. The World Wide Web Consortium was formed in September 1994, with a base at MIT in the USA, INRIA in France, and later also at Keio University in Japan. In Berners-Lee's own words, The Consortium

> "is a neutral open forum where companies and organizations to whom the future of the Web is important come to discuss and to agree on new common computer protocols. It has been a center for issue raising, design, and decision by consensus, and also a fascinating vantage point from to view that evolution." (Berners-Lee, 1998)

Since then the development of web technologies has been result of consensus between several kinds of interested parties being members of the consortium, both non-profit and commercial. This balance has prevented the fragmentation of the web, and made sure that the underlying basis for the web, the markup and the protocols to process it, would be free and openly available to anyone creating software for the web.

As we have seen, Berners-Lee, the creator of the initial World Wide Web had his own ideas and hopes for what the web could become. Inspired by both Bush and Engelbart, his initial ideas incorporated several of their ideas for what would make a good collaborative environment. As the web developed into what we have today, the problems discussed in this paper became apparent, and Berners-Lee and the consortium have as we will see, both created patches for instant solutions as well as making a basis for a future framework that will address the identified problems in order to create a possible future solution to them.

Chapter 2

The World Wide Web and HTML: What is wrong with hypertext at this point?

> "Trying to fix HTML is like trying to graft arms and legs onto a hamburger" (Nelson, 2005)

As we have seen, neither Internet nor Hypertext where new notions when Berners-Lee created the World Wide Web, and seeing that the Internet already consisted of many different channels of communication, he expected a lot of different types of data formats to exist on the web. Several hypertext systems existed, but all of them were created to work within their own limited environments, either made for a special type of computer or operating system. But they would not work together, because they were not interconnected, and did not talk the same hypertext, so Berners-Lee saw that some kind of hypertext "lingua franca" that any computer could understand would be needed if his hope for a universal collaborative system was to work. So he set out to create a simple hypertext language that would be able to provide basic hypertext navigation and simple documentation of the resources to be navigated to. In the original 1989 proposal for a distributed hypertext system, Berners-Lee noted that "generality and portability...[should be] more important than...complex extra facilities." (Berners-Lee, 1989). Generality and portability where probably the basis for the early success of the web (Cailliau and Ashman, 1999), but at the expense of leaving complex hypertextual facilities seen in older hypertext systems out of the specification.

SGML[1] was at that time, considered by the hypertext community as the only potential document standard, and was also the preferred language used by some of the world's documentation communities, which was the background for why Berners-Lee started creating his hypertext systems at all. Knowing that it would be difficult to encourage the documentation community to start using yet another documentation language, he chose to base HTML on SGML, creating an interchange format very similar to SGML. The HTML DTD [2] was added later by Dan Connolly, recognizing the importance of having a DTD to avoid the early web and early browsers from becoming fragmented due to many different non-interchangeable notations being created.

In (Cailliau and Ashman, 1999), Robert Cailliau one of the early pioneers working with Berners-Lee writes that ever since the early prototypes of what was to become the present World Wide Web, the hypertext community has pointed out shortcomings in the implementations of hypertext on the web. This resulted in the development of many alternative hypertext-enabled browsers, but all of them were application dependent, dependent on people seeking them out and using them. As we know, it was not the browsers with extended hypertextual functionalities that caught on and became used by everyone, and the aforementioned problems we are struggling with today may very well attribute some of their origins to the limited hypertextual functionalities present in the prevalent implementations. Other problems may be attributed to the specification and usage of HTML itself, but as we shall see, having defined specifications for good hypertextual features does not guarantee them becoming implemented in the mainstream browsers. In the following sections I shall identify some of the problems with our current implementation and presentation of hypertext and their causes.

2.1 The hyper in hypertext

The essence of hypertext lies in the notion that it should be possible to create branching points of departure in a document, making it possible to jump from one document to another one by a simple operation. This way it is possible to make seemingly arbitrary associations between things that might seem unrelated, just as the human mind works, something that to Vannevar Bush and Ted Nelson seemed to be what was a natural way of navigating and perceiving information. Looking back at experiences done in different hypertext implementations as in 1.4.2, and at the information overload phenomena identified in the introduction on the effects

[1] Standard Generalized Markup Language

[2] Document Type Definition, a set of rules defining what elements may exist in a markup and the allowed structure of those elements

of associative hypertext, it may seem that the nature of associative linking in itself, or our implementation of it is the source of some of our problems. Many hypermedia researches believe that disorientation and cognitive overhead are problems inherent in hypermedia, but it may also be that proper implementation of advanced hypertext features would help solve these problems. (Bieber et al., 1997)

2.1.1 Associative links

Bush lamented the artificiality of the systems of indexing and wanted to use associative linking to cross interdisciplinary barriers, creating a solution that allows us to navigate freely. But by doing this, we are at the same time losing other structures and navigational points usually found within a body of information. A hypertext link in the current prevalent implementation of hypertext, going from one point in one document to another point in an other document will really just point into a whole document containing the information the link points to, but within a larger, possibly different context. The link will not give any more specific information on where the relevant information is to be found, or other semantic clues like what the target of the link is really about or what relationship it has to the originating document or to the context it is situated in etc.

The only means available to content authors is the content of the link marker or the destination address resulting in links having very little meta information attached to them, often resulting in users not knowing enough about what the target of the link is about and where a link will take them, adding to the cognitive overhead of the user. (Bieber et al., 1997). What is of relative usefulness, is the possibility of using the <A>-element's "title" attribute to convey some description of the link and what it links to. This has, in the prevalent browser implementations, the effect of popping up a box holding that description if the users holds her mouse pointer over the link for a couple of seconds. This demands knowledge from the user about the description popping up though, and is regrettably inconsistently implemented by the prevalent browsers.

The possibility of using named anchors within documents will help us a little on the way, as the target page is scrolled by the browsers to position the named anchor at the top of the screen, taking us approximately to the right line within the target document. But the user might still be taken to another context where the association to the context of the point of departure is not always apparent. And even if the anchor's name is somewhat descriptive, and the user has the knowledge to extract it from the URL, the anchor's name does not necessarily carry any useful information such as to what relationship the target information has to the

information linking to it, and in what way the current context relates to the one one did a jump from.

Another problem is when the named anchors are in the same document, since the only means of judging the new location within the document is by the vertical scroll bar. With no extended information placed on the link, the user might not even know that she is still situated within the same context or page. Also the way the positioning of the information marked up by the named anchor is handled by different browsers, is sometimes confusing to the user due to the anchor not being highlighted in any way. And if a named anchor is close to the end of a page, the anchor cannot be positioned at the top of the screen making it very difficult even to determine where the author of the document intended to send the user. (Cailliau and Ashman, 1999)

2.1.2 Link directionality

Berners-Lee's original browser which he created locally on his NeXT computer included the authoring of links as distinct from the authoring of documents. This made it possible to create bidirectional links between documents, where a link always exists in the reverse direction. This feature together with link typing could give the advantage of showing the nature of the association between two documents at both sides of an association, as well as signaling to the user which role in an association the information of each side of the link plays. This could then provide the needed meta information to the links in order to lessen the degree of cognitive overhead caused by the user having to choose among several possible associations, as well as providing contextual clues lessening disorientation. The lack of link directionality on the web is caused by it being made up of resources with many different authors having different intentions for the function of their content. This is part of what ensures the generality of the web. Other hypertext systems having the possibility of creating bidirectional links usually would have an external link database, something which is not possible with HTML on the World Wide Web, since the links are usually static and embedded within the document itself.

2.1.3 Broken links

One problem identified already by Ted Nelson was that using addressable resources, one is bound to experience link rot. The current browsers will on receiving a numerical code of "404", which is an error code signaling that the resource on the specified address does not exist, replace the page of origin with an error

page displaying the error. This might cause a degree of disorientation for the user, alleviated by the "back button" having become a known navigational aid, taking the user back the one step required. This is a fundamental problem caused by the nature of the Internet, giving equal possibility to everybody to amend, delete or add resources having a fixed addressing scheme, and calls for a fundamental change if it is to change. Something similar of Nelson's idea of retaining multiple versions of all documents as discussed in 1.2.4 might be a possible solution, but calls for fundamental changes. What would be possible though, is a better error handling by the implementing browsers when receiving a "404" than to just present the user with the error code which in most cases is intelligible for the average user.

2.2 To structure or not to structure

When Bush lamented the artificiality of the current systems of indexing and called for associative linking he saw that finding information by browsing through hierarchies and structures was inefficient in his time, but he did not rule out indexing and structuring completely, as he actually discusses how future machines would be able to search through structures in a much more efficient way than a human could in his time. Ted Nelson is often interpreted to think that any structure is to be avoided, but instead he is in disfavor of all information having to be either sequentially or hierarchically structured, because it limits the kinds of structures that can be expressed.(Nelson, 2001). As we have seen, associative networks alone may cause information overload related problems if they are not enriched with meta information. So a solution would be to use both associative linking and structuring of information as we have seen some of the old hypertext pioneers do with their hypertext systems which would have both associative linking and some kind of ordered structuring principle applied to the information. Much of early research on hypertext and the resulting systems thereof has regarded having structuring principles besides associative linking as important. As Nelson has pointed out, not all content fits within the document paradigm, different types of information will call for different ways of structuring and structures might intertwine and overlap. It is then important to have a good and consistent way of expressing these structures which is both clear and simple and flexible at the same time.

2.2.1 Structure vs. Presentation

HTML in it self does not have many inherent constructs for creating structure as it is mainly following the traditional document structure of documents mainly used for print. There are 6 levels of headings, making it possible to structure an

HTML document in sections like this thesis. Then there are paragraphs, making it possible to divide content in manageable chunks, but they have no additional structural meaning part from that. There are numbered and bulleted lists, with the possibility of nesting levels, and there are tables with their columns and rows and header cells. In addition there are some logical elements having specific meanings making it possible to mark up a section of information with different functions, as with the <address> element, denoting that whatever is marked up by it is to be considered address data solely. Of these, there are not many useful structuring aids that could be used to diminish information overload problems, and much is left to either the browser or the author of documents to build structures through clever use of the tools available. Then, especially if a browser software is to make use of the meta information available, it is important that the few aids present in the markup are used correctly an consistently. This is sadly not the case though, an example being how the distinction between logical and presentational markup has been blurred.

In (Berners-Lee and Fischetti, 2000) Berners-Lee writes that he expected that most content would be within the resources linked to in the hypertext document, and did not foresee that HTML would become popular as a content carrier in itself, and probably not that it would become so popular as a presentation language. A basic design rule that guided HTML, he writes, was that it was supposed to convey the structure of a hypertext document, but no details of its presentation, since that was the only way of making it display reasonably on any of a wide variety of different computers, operating systems and screen resolutions.

2.2.2 Logical

Logical HTML elements, are elements that say something about the structure of the document, or that actually will add some extended meaning to the data enclosed within the element. When Berners-Lee first made HTML, he made no specifications for how data enclosed within these elements was to be rendered, that was later defined by the people who made the different browsers. So the fact that the element is rendered with a bold typeface probably made sense to the people who implemented the browsers, but on the semantical level it does not make sense at all. Semantically, the element tells us that whatever data found within the tags is to be considered as very important information, not that the data is supposed to be rendered with fat types.

The logical tags are not intended to carry layout information, and rendering them in a way that stands out from the rest of the data layout-wise, only under-

mines the argument that logic tags should not have any distinct presentational rendering, and also the argument that one should keep logical structure and presentation divided. When the different browser implementations in addition manage to render the presentation differently, it often leads to document authors compensating for the discrepancy between the browsers either by using a hack, or not using logical tags at all.

2.2.3 Presentational

Presentational HTML elements, are elements that are used to render the visual portions of HTML documents, such as images and tables, or otherwise elements that have a meaning that can be read through its visualization. An example would be the element that renders a piece of text enclosed within the element as bold, with fat types. As the element denotes **bold text**, the visualization of the element is rendered as **bold** text, a well known format known from literature and word processors. This time it actually makes sense when the browsers render the result as the element denotes.

2.2.4 Distinction blurred

To begin with, Berners-Lee was designing his hypertext system to be used by the documentation community and its need for a collaborative environment, he never intended HTML to be written by hand since he wanted the hypertext applications to be both browsers and editors. This way the integrity of the markup would be secured. To his surprise the human readbility and simplicity of the markup made it easy to write by hand too which made people start writing their code by hand. As discussed in 1.4.9, the WWW eventually would become interesting to commercial forces which eventually would impose their demands on the simple and clean markup that Berners-Lee initially created. As the early browser implementations having the most success by catching the public eye did the exact opposite of Berners-Lee's intentions by applying presentational effects on the logic markup, the exact things that Berners-Lee tried to avoid happened. Different browsers would implement these features differently, thus displaying the documents differently according to which computer, operating system or browser one used.

The problem that arises when a user sees that the logical element is rendered the same way as the element, is that the importance of making a distinction between logical and physical elements is lost. How can we expect an everyday user to care about which element she uses, when what is im-

portant to her is what the finished result looks like? The result often being that the user will use only the element even in situations where it would have been appropriate to use the element. The ultimate result is that documents on the WWW are so full of physical elements that the information itself drowns in a mass of presentational information, making it difficult to extract any semantically richer meta information from the existing markup.

Since making a distinction between logic and presentation has been important from the beginning, Berners-Lee and the world wide web consortium soon recognized what was happening when the distinction started to blur in documents found on the web. They knew that if it was to continue, the principle of generality of the web's implementation of hypertext was soon going to be lost, and they had to send the authors of hypertext the important message in some way. The first solution to this problem was the creation of Cascading Style Sheets, making it possible to separate logical and presentational data, and at the same time conveying the importance of this separation. In addition to help people understand that with presentational information stored outside of the document, what was really important was to use logical tags in the documents themselves, it also had the side effect of adding powerful means with which to manipulate the presentation of a document. Also, as CSS took hold and developed, and it was possible to access CSS attributes by the use of JavaScript, the main client side scripting language used for web pages, the document authors got better means of creating new navigational structures for their documents not possible with the the few structural aids already present in HTML itself.

Having clear and unambigous information structuring is of course important as an alternative to associative structures and having clear logical markup is then important to be able to express structures and making meta information easily accessable. Another reason why it is important to have clear and unambiguous logics and meta information in a markup is to provide for relevant searches in the information. Not everybody will browse for information following some structural scheme be it associative or otherwise, many will, if having the possibility, do searches in the material to be able to find information they are looking for. Researchers of information retrieval from Bush's time up until today have regarded being able to search information as valuable. As we will see in 3.3 it is then as important to make a clear distinction between logical and presentational markup as well as having good means of making metadata available.

2.2.5 Markup or Browser; where to place the blame?

The shortcomings we have been looking at, are mainly caused by those who implement the browsers, since as we have seen, the advanced hypertext features like the notion of link typing, adding semantical clues to hyperlinks has been considered as an important feature of hypertext since early on. Both Nelson (as seen in 1.2.4) and several of the other early hypertext systems as discussed in 1.4.4 saw the importance of typing links. Also, Berners-Lee, in his 1989 proposal for a distributed hypertext system introduced typed links as an "intriguing possibility" (Berners-Lee, 1989), and was included in in his proposal. Actually, HTML has always had a mechanism for specifying link types for relationships pertitent to the douments as a whole, through the *rel* and *rev* attributes of the <LINK> element. The relations were thought to be used in situations where one had several documents that belonged together in some sequence, where the "rel" attribute specifies a forward link, and the "rev" attribute contains its reverse relation. But since the specifications made by the World Wide Web consortium are mere proposals, this feature has been one of the ones being left out by the prevalent browsers.

Regarding structural elements making it possible to apply any other structures than the traditional document structure discussed above, the blame has to be put on the markup. It might be that Berners-Lee and the consortium regarded adding any structural elements to the markup as outside the function of hypertext, but there is reason to believe that the few structuring concepts found within HTML are not adequate when wanting to express other structures not belonging to the document paradigm. It is of course very much up to the author how she chooses to use the elements already present in HTML to create new structures, which is of course possible, especially with the use of additional technologies such as CSS, JavaScript and Java etc, but if one wants to reuse the code for other purposes basing the reuse and restructuring on the meta information found in the structural elements, HTML is not adequate.

2.3 A parallel

A helpful parallel to see why our present hypertext implementation is badly designed is a look to what happened to programming languages in the 1960's. In (Karabeg et al., 2005), the authors tell us how computer programs written before the birth of high-level programming languages resulted in thousands of lines of what they call "spaghetti code", called so because of their spaghetti-like structure. It was the generic "goto" program control that caused these problems because it allowed for arbitrary jumps from one context to another within the code. As with

these programs, hyperlinks are like "GOTOs" too, just pointing arbitrarily to another context resulting in a "spaghetti code" of the internet. A further parallel is when hypertexts get big or lacking good structuring or meta data, the ways of handling things which were created for smaller volumes no longer works for larger ones. The solution to the problem of programming languages was abstraction and modular organization through creating high level structures and modelling methodologies for how one ought to structure information into manageable modules. The parallel to associative linking is clear and points out why hyperlinks as the only means of navigating information does not work very well for our needs, and maybe also what could have to be one possible solution to the information overload problems we are experiencing.

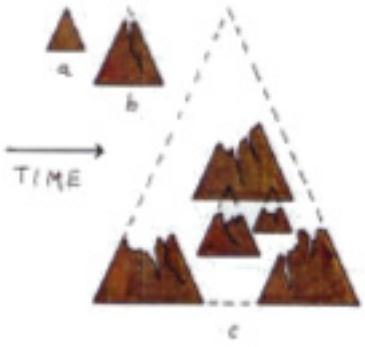
2.4 High level information and Polyscopy to combat information overload?

In (Guescini et al., 2005), the authors propose an information design methodology believed to combat the information overload problems experienced with not only the present hypertext but informing in general. Pure associative linking alone as an information structuring approach is considered as inadequate for our present needs, and the conscious design of structured information in terms of small manageable modules, created on different structural levels is seen as a possible solution. Polyscopic modeling is based on the notion that the scope or the perspective determines the view, or that our way of looking and communicating detrmines what we are able to see and express. This is similar to Douglas Engelbart's "viewing filters" and Goldstein and Bobrow's "perspectives" in their "Personal Information Enviroment" hypertext system mentioned in 1.4.3. To visualize the point of poly (many) scopes, the authors use the metaphor of the mountain, where every point of outlook represents a single viewpoint or scope of the information available at a time. Information is thus thought to be consciously designed into modules representing a single view.

Bush was lamenting the fragmentation of information due to specialization, and how it was difficult for somebody living on such an "island" to have access to information being produced on other islands. The authors uses this as a metaphor as seen in fig. 2.1, of our present way of informing also, not saying that our information is fragmented caused by having too many information islands without any connections to eachother, beacause with the possibility of associative linking that is no longer a problem. Instead it is suggested that the fragmentation is due to having too litle "high level information", meaning that each island is lack-

ing high level information giving us an overview of the information contained on the “island”. It is believed by the authors that by providing high level modules representing the fragmented informational islands it is easier to get an overview, allowing us to make sense out of them. Systems like “Hyperties” and Brown’s “Intermedia” project discussed in 1.4.6 identified much the same need for high level information, giving the user high level overviews of vast amounts of information, reducing the risc of getting “lost in hyperspace”.

Figure 2.1: Information fragmentation

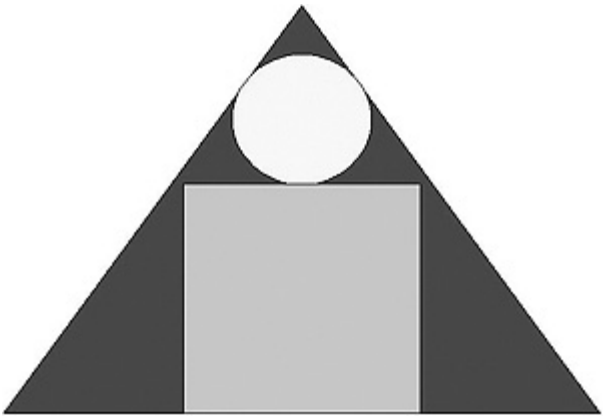


Information is thought to be structured on several dimensions, not only in high level modules by scope. The ideogram in fig. 2.2, reflects the design of Polyscopic Information as a whole consisting of distinct parts, where the “high-level” part is represented by the circle and the “low-level” part is represented by the square. Polyscopic Modeling proposes three kinds of abstraction for organizing information into modules:

The vertical abstraction can be understood as metaphorically climbing up or down a mountain providing lesser og higher degrees of granularity or detialied information. The vertical abstraction is thought to be symbolized by the circle, involving the rounding off of details, presenting the high level, main point.

The horizontal abstraction is symbolized by the square having distinct sides representing scopes or angles of looking. The square represents “low level” information, but having scopes gives the possibility of seeing as simple perspectives of the more detailed low level information as possible.

Figure 2.2: The Polyscopic Information Ideogram



The structural abstraction is symbolized by the triangle holding the high level and low level modules together as a whole. While the vertical and the horisontal abstractions suggest that information should be created as modules, the structural abstraction respresents the relationships between these modules and allows us to put them together in to a whole. The main point behind the structural abstraction is that structure is an essential property of information and is necessary to bind our fragmented information together. It is the structural abstraction that makes the notion of polyscopic navigation possible implementing the principle that the scope should determine the view.

What the Polyscopic Information Modeling suggests is that to be able to perceive information efficiently as we need it with the amount of information on the web growing every day, we need to see information from the top of the mountain. The current way hypertext is implemented on the web, has not solved what Bush suggested associative structuring could do, it still gives us more information than we can handle resulting in the information overload related problems discussed in the current thesis. That is why the vast amount of detailed information must be abstracted and consciously designed and structured in ways that will lessen information overload.

2.5 If not HTML, then what?

Looking back at several of the the old hypertext projects, most notably Engelbart's NLS system, it is suggested that modelling information in modules and

levels as well as having typed associative linking to access the information modules on different levels may be a better way of structuring information as hypertext than the current implementation of hypertext, giving greater opportunities to combat information overload related problems. Bush's wish that we could be able to connect fragmented information by association has been fulfilled by our current implementation of hypertext. People are now able to surf the web associatively to seek out information much closer to what Bush believed was natural to the human mind, but as we have seen, that is not enough. It is important that information is structured by other means than pure association in order for us to be able to make better use of the information.

In the following chapters I will look at prerequisites for transcending the limitations of HTML and new approaches to hypertext on the web. They are either markup based solutions reflecting the shortcomings of HTML, or they are schemes for how metadata can be used to change the way we structure and access information on the web. Some of the new solutions are future considerations demanding quite fundamental changes, thus, this thesis will also take into consideration what could be done by using some of the new technologies on basis of existing resources to cure the ailments of the web for the individual user. The possibility remaining since we can't change the web or HTML at this point, is an approach where a new type of browser could be imagined implementing some of the needed advanced hypertextual functionalities discussed in earlier parts of the thesis by implementing Bush's notion of the trail and the Memex.

Chapter 3

Metadata

The word “meta” is Greek and means “with” or “after”. In modern context “meta” is used about concepts which are transcendent, something that exists as a addition to, together with, or over the original concept. The word indicates a higher level than what is concrete, something abstract. The word data would denote whatever information or facts are constituted of. In the context of information technology, data would be anything that can be saved into a machine-readable format. So whatever is saved and read by a computer is thus data. (Borum, 2003)

Metadata is "data about data", or high level data describing other data, thus transforming the data in question into something closer to what we might call information by adding meaning to the data in question. For a human being, metadata would be the sum of knowledge we have within a subject domain for a piece of data, which helps us make a sensible informational unit out of it. Metadata is not a new phenomenon that was born with the use of the World Wide Web, library science has for a long time made use of metadata in the cataloging and indexing of information to create, describe and improve access to an object.(Borum, 2003)

3.1 Processing information : Human vs. computers

Computers are in essence just very big and powerful calculators, and thus do not know anything more than just the input they are fed. So if you feed a computer with a piece of data it does not know anything more about it than that there is a piece of data occupying some part of its resources. If you tell it to process the data, it will only be able to process whatever you have fed it with, so if you just told it that there exists some data and the computer stored it in its own native binary language format, the only thing it is able to do with it, is to return it to you upon enquiry.

Now if you tell the computer that there exists a piece of data, and that the data for instance is a picture, and that pictures are supposed to be treated in a specific way, the computer will react accordingly, it will figure out that since the piece of data occupying its resources is a type of data that should be treated in a certain way it has to turn on the parts of its resources that make these expressions become possible. It cannot do anything past the instructions given to it.

One could of course think that as a human being one would not have the same problems as a computer in making sense out of a piece of data. And it is partly true, we are of course capable of making far better inferences on the background of some piece of data than a computer is able to. Since we are always updated with contextual clues in our daily lives, we do have a much greater frame of reference that we can use when making assumptions to what the meaning of a piece of data would be. But mostly, we are not in a very different situation from computers, since we really have to be "programmed" with the additional information we need to make sense of a piece of data. We can still experience the same problems found with computers, such as figuring out what would be the correct meaning of a semantically ambiguous piece of data. If we are presented with a piece of data that we do not have any specific knowledge about, nor have any sufficient contextual clues for, we will look to our more general knowledge of things and try to infer a semantical value for the data in question. This of course will often lead to ambiguities, and difficulties in processing the piece of data correctly. In situations when we have no clues to what the data in question is about, we are left in much the same situation as the computer, the difference being though, that we do have the autonomous will and possibility to go further and search for or learn about the possible semantical value of the data in question.

3.2 Why do we need metadata on the web?

As I have discussed earlier in this thesis, we have a lot of dispersed information resources spread on the Internet. And much of the information found on the World Wide Web is often so badly marked up or ambiguous regarding its context or its meaning that we could rather just call it data. Data in the sense that if you fed a computer with it, it would not be able to do any more sensible operations with it than just presenting it as it is. In order to call something information one would have to attach additional meaning to the data in question, making it possible to consider the the data in light of the metadata. A word or piece of data becomes intelligible from its context or other semantic clues, and thus it is necessary to add metadata to the data in question in order for it to have a computable meaning in case it is being consumed by an agent accessing it from outside the original

domain or context. With metadata in place we have better tools with which to make inferences about the data in order to do more interesting tasks with the information than just to render it as one would with a traditional document format.

3.3 What to use metadata for

Within our context of information overload on the World Wide Web, it is of great use to be able to give a browser, or whatever software which is processing some data some clues on how the data is supposed to be processed. One apparent use already discussed in the former sections is using metadata to provide richer information to hyperlinks by providing typing, descriptions etc. Other uses one could imagine, would be to attach metadata to portions of content making it possible to group content in more useful ways than by headers and paragraphs as done in HTML, as a result, making it possible to create better navigational aids or filtering mechanisms being one part of combating information overload. However, the most known use of metadata on the web up until now, has been to support information retrieval in search engines.

The best known vocabulary for metadata is Dublin Core, which is a set of 13 properties that may be applied to information resources to describe them and to support information retrieval. It is independent of any particular technology (Garshol, 2004) and can thus be applied to any informational resource. They describe the typical things one thought people would want to search for when searching for information on the web. Examples are:

Title The title of the document

Creator The author of the document

Publisher Who published it?

Date When was it created?

In (Garshol, 2004), Garshol points out that one has to figure out what kind of information about the data would help the user most when trying to find whatever she is looking for. He says that one common case would be that the user has seen an informational object from a previous search, and remembers specific details about it, such as the meta data described above and could form her new search on basis of those meta data. But the question remains; is that really helpful in most cases?

Garshol goes on saying that it is clear that the type of information listed above does not really help the user in establishing what a document is really about, part from maybe the "Title" attribute that might provide the user with some clues on what the document is about, but does not necessarily mention all the words the user really needs to figure out if the document is relevant. He also says that the title attribute sometimes might even presuppose knowledge the user does not possess, having the result that the user might be loosing out on valuable information that she could have had if she understood what the document actually was about. The *Subject* meta data element on the other hand is different, as it gives the possibility to say something about what the document in general is *about*. Garshol points out that the difference in semantical power between the Subject element and the other ones, shows us that standard metadata mostly provides administrative information, and that it says very little about the *subject* of an object, and that out of the Dublin Core metadata properties only *title*, *description*, and *subject* will help the user in establishing what the document is about.

The HTML language itself is not specifically loaded with metadata, it just has the structural levels found in traditional documents, like headers, paragraphs, tables etc. To provide for meta information, it has the <META> elements, designed to assign additional metadata to the overall document itself. These meta elements enable us to define a whole range of properties, including some of the same properties defined in Dublin Core, such as "Author", "Publisher",and "Date" among others.

There are many more properties that can be defined using the <META> element, but not all of them are standardized, as the HTML DTD does not define a set of legal meta data properties. The idea was to have the meaning of properties and the set of legal values for it to be defined in a reference lexicon called a "profile", where for instance a profile could be designed to help search engines index documents. This profile was to be referenced by the "profile" attribute of the <HEAD> element of an HTML document. However, most or all of these properties will usually just say something about the overall document itself.

The meta element attribute in HTML which has been of some use regarding the content of a document, is the concept of keywords, which one would use to enumerate information on what subject or subjects the document is about. Search engines , or *web crawlers* indexing the World Wide Web were supposed to index web pages on basis of these keywords together with other properties extracted from the meta elements. And in the early beginnings of the World Wide Web, the indexing software of search engines would use these keywords to help determine

how to rank documents in their returned search results. But as time went by, the keyword meta elements were corrupted by document authors abusing the meta element attributes with different schemes designed to attract attention they would not have gotten otherwise. One approach has been to repeat the same keywords several times in an attempt to get a higher ranking on relevancy rankings, an activity called *spamdexing*. Another similar one was to add information that was not proper for the content of the page, but which would attract visitors. (Wikipedia, 2005b). This of course led in the end to a situation where search engine algorithms were programmed to ignore keyword information, either partially or completely by giving less or no weight at all to the keyword metadata assigned to these documents. Today search engines give most weight to actual content found within the documents and will index on basis of that. Some engines will still index the meta keywords, but will just use them as a minor supplement to the content and the title elements found within the documents. Some search engines will even detect what they suspect as spamdexing and remove these pages from their indexes. Indexing on basis of content will suffer from many of the information overload related problems identified with HTML, since there are few or no clues to what the content is about, often resulting in ten, maybe hundreds of thousands of irrelevant hits.

There are legal techniques used to improve your documents ranking with a search engine. This approach is called *Search engine optimization* (SEO), which is a set of methodologies aimed at improving the ranking of a website.(Wikipedia, 2005a). But as this became prevalent too, the creators of search engine ranking algorithms had to adapt their algorithms again to be able to return the most relevant listings, rather than the cleverly optimized ones. Most of the spamdexing and SEO activity is mainly performed by webmasters promoting commercial web sites, something which regrettably results in the degradation of information found in search engines, leading to less relevant hits resulting from a search.

The search engines of today uses different indexing techniques, Google for instance, has since 2001 had success with its PageRank feature, where a web page is given a rank based on how many other web pages links to it on the premise that good or desirable pages are linked to more than others. In (Henzinger, 2005), Monika Henzinger from google.com lists two assumptions made as the base for this approach.

Assumption 1: A hyperlink from page A to page B is a recommendation of B by the author of A.

Assumption 2: If page A contains a hyperlink to page B then the two pages are on related topics

Henzinger, adds that Assumption 1 does not hold for all hyperlinks, but that it still seems to be "close enough" to the truth so that techniques that are based on it work well in practice. In fact, Google has had great success with this approach which is a good step in direction of more relevant hits. Henzinger attributes the strength of this approach to the fact that the ranking is dependent on the content of other pages than the document being ranked, and thus not susceptible to be controlled by the author of the document. However as there is no such thing as a fail-safe algorithm, webmasters have found a way round to optimize their document ranking also within this scheme. More fundamental is the fact that the search result is still dependent on doing a keyword search based on what the user entered, and finding pages that had matching keywords within the document body. Even if some of the relevancy problems have improved it is really just on the surface, since hypertext links are not typed, and data within a document have little or no metadata, search engines like Google will still suffer from overload caused by problems like keyword synonyms etc. creating ambiguity.

Yahoo!, started up as a subject-based directory of links, where they actually have humans indexing pages and creating taxonomies as navigable directories. The directory function is in many ways a step in the right direction towards giving data a relevant context, giving the user a better clue to what the document is about. The indexing being performed by humans, being able to assess the relevance of the content within a document to a much greater degree than a more generic software algorithm would, helps considerably in placing web pages in an accordingly relevant subject category. On the Yahoo help page, the answer to the question "How does the Yahoo! Directory differ from Yahoo search", is that the Directory listing might be helpful when you are not sure how to describe what you're after, or you are searching for words with multiple meanings. (help.Yahoo!.com, 2006). The search engines have gone a long way in creating advanced and clever algorithms to be able to make something meaningful out of the messy document contents and the poor meta data usually found within HTML documents, however, is not adequate by itself as a real alternative to associative browsing or as a sole solution to our information overload problems.

3.4 Meta, but how to create it?

We know then, that we need more and better meta information, and that our old ways of creating it is just not working as intended. One thing is that the

syntax for expressing metadata in HTML is not expressive enough, but the meta data we have access to is much to fragile and susceptible to errors due to human factors such as the problems discussed above where authors of documents corrupt the metadata for short-sighted personal advances, but also simpler problems such as misspelling, or authors using different keywords for the same content etc. Berners-Lee probably saw that some of these problems were inevitable when he wanted the initial browser implementations to implement the possibility of editing. That way, at the same time as one created the means for having a collaborative environment on the web, one could also make sure that the markup would keep its integrity by having the software generate it correctly. When that did not happen, the importance of creating a DTD for HTML to ensure the integrity and uniformity of both markups an browsers was apparent. But no validating browsers were created which could have ensured some degree of integrity on the markup or the metadata, as generality and the freedom for anyone to publish , and to have the web growing was considered more important at that time. Instead, commercial forces creating the prevalent browsers would follow public demand for presentational power, as well as becoming very loose on their interpretation of faulty markup. This again had the effect of sending a message to many authors and users that "anything goes", creating a vicious circle of bad markup leading to browsers becoming even sloppier on validation again leading to more bad markup and so on. The ultimate downside being of course, that the integrity of the HTML being produced became even less suited to combat the ever increasing problems of information overload. So how is an author supposed to understand that concepts such as the distinction between logical and presentational elements, the division of logics and presentation, the integrity of metadata etc. are important when the prevalent implementations and practices are so focused on presentation and short-term gratification?

This tricky problem was evident very early in the history of the World Wide Web, and by the mid-1990's some people at The World Wide Web Consortium seeing the character of the growing world wide web and its emerging problems, believed that looking back to SGML would offer some solutions to the evident problems the WWW was going to face as it grew. (Wikipedia, 2005d). What was needed, they thought, was a new markup language that was semantically more expressive than HTML in order to address some of the information overload problems discussed in the previous sections. But also a language that was strict enough to disallow all the faulty code being written, making the code ambiguous for machine processing at the same time as being flexible enough to ensure the generality of the web.

Chapter 4

New Markup Languages for a new era

4.1 Standardization

Standardization often means making a set of rules for how something is supposed to be done in order to ensure that one can expect the same result every time one uses the standard in question. Strict standardization of HTML was sacrificed to promote the growth of the web, and has shown itself to not be useful as an acceptable standard, and is also too limited in its semantic expressivity to be able to cater for the needs of today's and the future's informational requirements. So what had to be done, was to find a way of making a language that at the same time as it had to follow and conform to certain standards, it was not limited to a simple set of elements, limiting its use for a broader domain of usage than just producing simple documents for human consumption. Standardization is of special importance when we want to make documents that are supposed to be machine processable, and since computers are not very good at doing anything that it does not know how to do in advance, we have to ensure that everything being fed to the computer is conforming to a format that the computer expects and knows how to process.

4.2 XML, the eXtensible Markup Language

The reason the creators of XML looked to SGML for inspiration, was that at the same time as it was a quite strict language, it was a meta language, making it possible to define new languages from it that would still have to follow certain rules of conformance. HTML is a language defined by a SGML Document Type Definition (DTD), and was intended to be a standard which was to be in-

terpreted according to some of SGML's strict rules of conformance, something which sadly didn't happen. So instead of trying to define another language by using SGML rules, a group consisting of Jon Bosak, Tim Bray, C. M. Sperberg-McQueen, James Clark and several others, began the work on a "light" version of SGML in 1996. As a result, the XML 1.0 specification was adopted by W3C as a new standard in 1998. What makes XML a suitable language for standardization, is that it has to follow a set of rules to be considered a valid document within the standard. This ensures that every XML document has to follow certain structural rules, making it much simpler to process its data.

4.2.1 Checking documents for Well-Formedness

Since XML is a subset of SGML, it has to adhere to a basic set of rules which applies to most markup languages, including HTML. And any application which parses XML is required to report an error if the XML file does not conform to these rules. The rules ensure that important requirements for unambiguous machine-processing are met, such as the closing matching start-tags with end-tags to ensure that the computer understands what to assign metadata to. Other important requirements are proper nesting of elements, proper quoting of attribute values, escaping of characters that may be ambiguously interpreted etc. There are several more things to take in consideration that may cause a document to be malformed, but these are the most important ones that have to be observed (Harold and Means, 2002a) since they ensure unambiguous parsing as well as a minimum of standardization.

4.2.2 Checking documents for validity

One of the rules of well-formedness mentions the Document Type Declaration, which is a set of rules defining which elements are allowed within a document, and in what order and quantity they may appear. The power of XML lies within the scope of the DTD, since an XML document may or may not adhere to a DTD defining it. An XML document may be written without the need of any rules, which will be sufficient for situations where it is not important that the document conforms to a standard, as with personal use, or at least within a controlled environment where the XML does not have to be parsed by a computer program which has to accept document instances from several sources. On these occasions it only has to be well formed to be regarded as XML. Validity on the other hand demands that the document conforms in every aspect to its DTD, this way, a validating parser can compare the XML document to its DTD and report errors if the document does not conform to the DTD. This would be very important in situations where XML-documents are produced by many different people and is

required to be processed by i.e some piece of software that expects the data to be tagged with a specific format. This way there will be no ambiguities that causes the software to process the document incorrectly or not to process it at all.

4.3 XML, the future document format?

XML meets both requirements needed for our modern informational needs, it is both machine-processable and simple enough for human consumption, meaning that if one gave an XML file to a human, she would be able to figure out the meaning of the elements, and thereby be able to create software to process the XML according to her own needs. Since one has the possibility of creating whatever elements one needs for the task at hand, it also gives the user means to create a document format that is much more flexible than using i.e HTML which follows a traditional document format not necessarily adapted or flexible enough for our present and future informational needs.

4.3.1 XML for Web documents

When it comes to catering for the visual needs of humans when creating web documents, XML in itself is not immediately there. Since the authors of XML wanted to separate structure and layout, XML has no inherent visual rendering if shown in a web browser, it will only be rendered as plain text, or some browsers will even render the elements themselves, making it possible to navigate the structure made up by the nested elements. The reason for this is that XML has no set of predefined elements like HTML, and it is therefore not possible for a web browser to visualize the document without an additional style sheet document written in either Cascading Style Sheets (CSS), or in the eXtensible Style Sheet Language (XSL), a style sheet language defined in terms of XML describing the layout for the specific set of elements present in the XML file. Thus it truly is an embodiment of the idea that logics and presentation are to be separated.

The future for web documents lies in the use of XML, but since it still is a bit cumbersome for the average user, W3C saw that a transition was needed, so to ease the transition three new DTDs were introduced, making up a set of rules for XHTML, the eXtensible Hyper Text Markup Language, which is what we could call an XML-enabled HTML, giving the possibility of embedding XML within the HTML and to use XML tools like XSL to process the code. When it comes to validation, nothing has really changed though, because since it is still HTML, the web browsers still will interpret the document according to the same rules as they have for traditional HTML. Of course one *can* validate the code, as was always

the meaning even with HTML, and W3C has set up a validator for markups that will help the user validate their documents, but the ones that will validate their code will mainly be the same ones that already validated their HTML to begin with. Still the conceptual change is important, it shows the direction in which the production of documents for the WWW will go, towards standardization and validation, and gives the possibility to warm up to the changes by following the transition. This is all reflected in the creation of the three DTDs, which are named Strict, Transitional, and Frameset, where the Frameset DTD is the same as Transitional but specific to framesets. The Transitional DTD allows for creation of HTML documents with very few amendments from the old ways of HTML, while the Strict DTD is much more restricted and focuses on real division between structure and presentation, giving authors a way of easing into future ways of producing content.

When the browsers of the future shift their attention to XML, the story will be another. Already, the newest versions of the main browsers has a fairly good XML validating parser implemented, which will root out both well-formedness errors as well as validation errors, making it impossible to have your document rendered unless it conforms to the standards. The author of documents is then forced to conform to the standards, which will ensure standardization on the web and ensure machine readability. What remains to be seen is how it is going to affect the content produced by humans. Most computer programmers are better trained in working with the rigid structures one encounters in data-oriented applications than in the more free-form environment of an article or a story, while most writers are more accustomed to the more free-form format of a book, story, or article. (Harold and Means, 2002b). It might even be that the time for Berners-Lee's vision for browsers with editing capabilities has come, creating a more human friendly environment for the production of content, while ensuring that that the resulting documents will meet the rigidness demanded by computers.

With XML, several auxiliary XML-based languages have been born, XSL being one of them ensuring that we still will be able to create layout for our hypertextual presentations on the net, but also languages that will help to implement some of the advanced hypertextual features discussed in earlier sections. For instance, the possibility of creating advanced linking abilities are suddenly much closer with the creation of XLink, XPath and XPointer. XLink's linking model gives the possibility of creating typed links as well as the possibility of adding roles to be played by the sources in the relationship represented by the link. Other possibilities are multiple locators to represent multi-ended links, traversal behaviors,

the possibility of distinguishing between different sources like whole documents, nodes, sections etc..

XPath and XPointer are languages created to be able to target any element in a document, making it possible to address sections within a document which is much more fine-grained than HTML's anchor system. Also, with HTML if someone else but the author of an HTML document wishes to link to some section within a whole resource, she has to rely on the author of the resource having defined anchors in advance. This is changed with XPath and XPointer, as one can locate any node, point or selection in an SGML, HTML or XML document.

4.3.2 XML for other media

If we wanted to create information for some other media than the World Wide Web seen through a web browser, XML is also the right tool to use. Since the birth of HTML there are many new media that have been born that are wired up to either the World Wide Web or Internet in general that can read electronic documents. These devices will often have their own way of visualizing information, and will therefore need another format than HTML, which is neither strict enough for machine processing nor flexible enough to work for several media. Since XML is a strictly logical markup and not being burdened with any presentational limitations, one XML file could serve as a basis for delivering information to many different media at once, you only have to know the expected format of the target and create a fitting processing software that will transform the data into information fitted for it.

This is perfect for computerized applications, but what about the human aspect? On would expect that humans would produce some part of all the content delivered to the different media. This thought has been taken into the newer revisions of XHTML, trying to create a middle ground between the strictness of XML, and the need to let humans produce content as hassle-free as possible. As of XHTML 1.1 the concept of modularization was introduced. XHTML Modularization is a decomposition of XHTML 1.0 into a collection of abstract modules that provide specific types of functionality. These modules may be combined with each other and with other modules to create an XHTML subset and extension document types that qualify as members of the XHTML-family document types.(Consortium, 2001). This enables the user to write her XHTML as she used to, but also to add distinct XML modules that will provide content tailored for other media than the web browser on the World Wide Web. Each application, be

it a web browser or a mobile telephone, will pick its preferred markup and ignore the part of the markup it does not understand.

4.4 XML based markups

Since XML gives the possibility for anyone to make a new markup, several markups have seen the light of day. Most markups are just used locally within intranets or somewhat closed communities, or even by just a few persons, but some markups have been published and are widely used, and some have even become standards within their domains of application. Known examples are:

MathML an XML data format for presenting and capturing Mathematics on the web. It is a W3C recommendation.

InkML an XML data format for representing digital ink data that is input with an electronic pen or stylus. It is a W3C recommendation.

SOAP The Simple Object Access Protocol, an XML data format for exchanging structured and typed information between computers in a decentralized, distributed environment. It is a W3C Recommendation.

SVG a language for describing two-dimensional graphics and graphical applications in XML. It is a W3C recommendation.

CML Chemical Markup Language, an XML data format for presenting and capturing Chemical information.

There are of course many more, but this list illustrates the applicability of XML, bringing new potential to the World Wide Web, Internet and machine processing of information.

4.5 The new era of the Semantic Web

With the new possibility of creating machine processable metadata, other possibilities open. What if we could have machines do the searching, browsing and reasoning for us? Berners-Lee and the World Wide Web consortium has recognized the amount of information living on the World Wide Web that is not strictly meant for human processing, which with sufficiently good metadata assigned to it could perfectly well be transformed into content which can be processed and understood by machines. As Berners-Lee puts it in an interview in June 2005, “It’s about creating things from data you’ve compiled yourself, combining it with

volumes (think databases, not so much individual documents) of data from other sources to make new discoveries." It's about the ability to use, reuse and repurpose vast amounts of data which currently is in relational databases, XML documents, spreadsheets and proprietary data files, all of which would be useful to have access to as one huge database.

In (Bush, 1991), Vannevar Bush looks back at his now famous 1945 article "As we may think" and in the light of the new technology at the time of writing, he adds the idea of automatic production of trails to his Memex. He envisions the Memex as being able to do inferences on background of clues it has learned from the user's actions. The user could then give the Memex instructions by giving it appropriate arguments and let it do a search in its available material, thus creating a trail by itself. The parallel is not accurate but is useful when trying to imagine what a future Semantic Web could be like, because it is not supposed to replace the existing web of HTML documents, as an automated trail-making function in Memex never could replace the function of the personally designed trail. The Semantic Web is not about the meaning of the existing documents, or marking up the existing HTML documents to let a computer understand the content, instead the point of the Semantic Web is in the potential for new uses of data on the Web. It is an addition to the web, not a replacement of it, but browsers may have to reinvent themselves though, not only to be able to process XML better, but also to create new functions for the Semantic Web. Possibilities are many, in the 2005 interview, Berners-Lee envisions the possibility of browsers using Semantic Web data to accompany human-oriented resources, using the Semantic web metadata to select and marshal human-oriented metadata, but also to have data centered display types resembling current applications made to display and administer relational databases.

As we have seen in the sections about metadata, it is difficult to rely on humans creating and encoding web content consistently, and to have them see the importance of creating good meta data. The semantic web is not dependent on that, what it needs is for more and more applications generating machine-readable data. To make that happen, the World Wide Web Consortium has defined Semantic Web standards for the description of data and to describe the associations between the data resources. These are some of the prerequisites needed for the web to make the transition from what i call "knowledge presentation" to "knowledge representation", where the focus shifts from presenting content to representing concepts, going from a document-centric to a subject centric web. In the following sections I will look into prerequisites for subject centric knowledge representation and different approaches to it.

Chapter 5

The Resource Description Framework, and the Semantic Web

In (Powers, 2003), the description of RDF starts like this:

> "The Resource Description Framework (RDF) is a language designed to support the Semantic Web, in much the same way that HTML is the language that helped initiate the original Web. RDF is a framework for supporting resource description, or metadata (data about data), for the Web. RDF provides common structures that can be used for interoperable XML data exchange."

As the quote says, RDF is a framework supporting resource description, existing independently of any representation, but it is usually expressed, or serialized as RDF / XML, which is the recommended serialization technique for interchange. It is also the main effort from W3C's part to solve the information overload problems identified in the earlier sections, being the cornerstone of their goal of creating what they call the Semantic Web. W3C's RDF primer (Manola and Miller, 2004) says that RDF is a language made for representing metadata about Web resources, such as title, author, and modification dates of a Web page, copyright and licensing, or the avilability schedule for some shared resources. But not only that, RDF can also be used to represent information about things *identified* by the web even when they cannot be directly retrieved. People, places and things in the physical world can be referred to as well using URI's, which are more general than URL's and thus do not suffer from the inherent unstability of URL's. Other possibilities of identifying subjects is for instance by a persons email address for a person, or by the address to a webpage about some subject. RDF is intended for situations in which information needs to be processed by applications, and made available to them without loss of meaning, rather than being only displayed to

people. Meaning also that it is intended as a language for interchange between applications.

RDF is based then, on the idea of identifying things using Uniform Resource Identifiers, which are unique text strings identifying a resource on the internet uniquely. As soon as a resource is identified, it can be made part of computations, and RDF is a means of describing these resources in terms of simple properties and their values. These descriptions enable us to make simple statements about the resources in the form of graphs, containing nodes representing the resources, and directed arcs, representing relationships between the resource nodes, their properties and their values. This enables us to create statements much like statements in Description Logic, a branch of First Order Logic. Let us say that we wanted to convey the meaning of the follwing sentence to the computer: "XML tags information". If we wanted to say the same thing in RDF it would look like this:

<rdf:Description about="[XML]" xmlns:my="[my]">
<my:tags rdf:resource="[information]">

A resource is anything that has identity. The [XML] and [information] are supposed to be URI's and are therefore resources, and gets "identity" from whatever the URI is pointing to. The order of the resources in the sentence will tell which of them plays the role of Subject and Object. The above statement is equal to the syntactic tree seen in 5.1:

The RDF data model is as mentioned above, a graph; a mathematical construct that connects nodes and arcs. These graphs are collections of statements called "triples" as seen inf fig. 5.2

As we can see, the RDF statement is made up of tree parts which forms a triple of

- Subject
- Predicate
- Object

This framework gives us the ability to teach computers to make statements on its basis. These graphs are very flexible, and we can of course connect multiple statements in one graph:

Figure 5.1: Syntactic tree

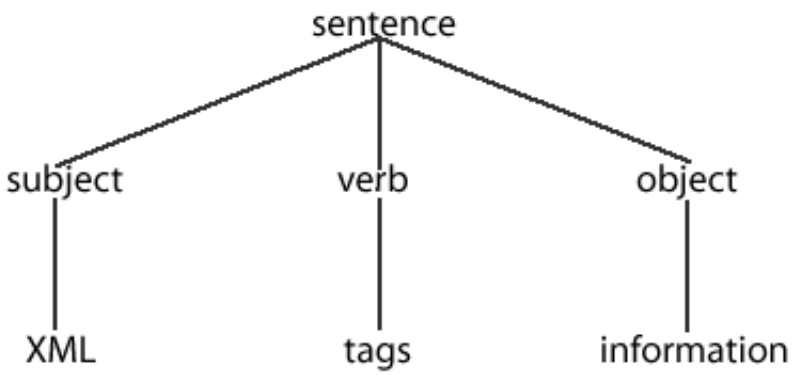


Figure 5.2: The direction of the arc tells us what resource is the subject or the object.

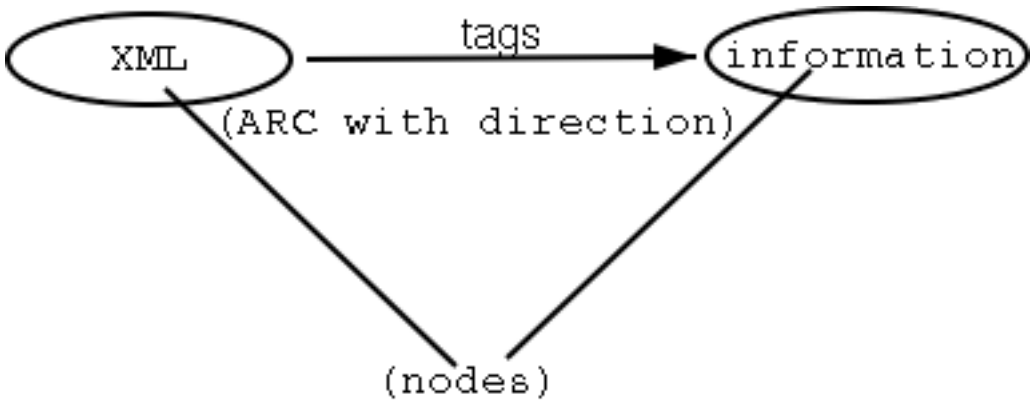
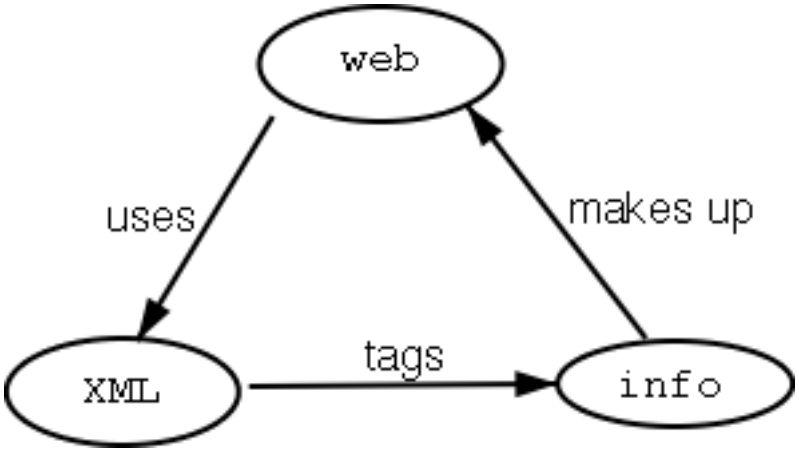


Figure 5.3: Multiple RDF statements.



What happens, is that a resource then can be the subject of one statement, but the object of another. This way it is possible to express quite advanced structures, since any resource may participate in an infinte number of statements. Since the structure is made up of subjects, objects and predicates, this maps well to **First Order Logic**, which can be used to compute inferences on basis of the RDF triples, through the use of FOL's richer syntax implemented by some software agent..

5.1 RDF vs. XML

When it comes to *Semantic interoperability*, RDF has significant advantages over XML. Semantic units are given naturally through its object-attribute structure, and all objects are independent entities. Also, RDF describes an abstract model independent of XML, the RDF model can still be used, even if the syntax is changed or disappears. Plain XML on the other hand, has disadvantages when it comes to semantic interoperability. XML is aiming at the structure of documents and does not impose any common interpretation of the data contained within the document. The major limitation of XML within this context is that since an XML expression has no inherent semantics, its semantics is only determined by the actions that one or more programs undertake on the XML expression. An RDF expression on the other hand, has specific declarative semantics, and this is specified independently of any processor for RDF expressions. XML in itself is very suitable for data interchange between applications that both know about what the data is, but not in situations where the addition of new communication partners occur frequently. (Decker et al., 2000)

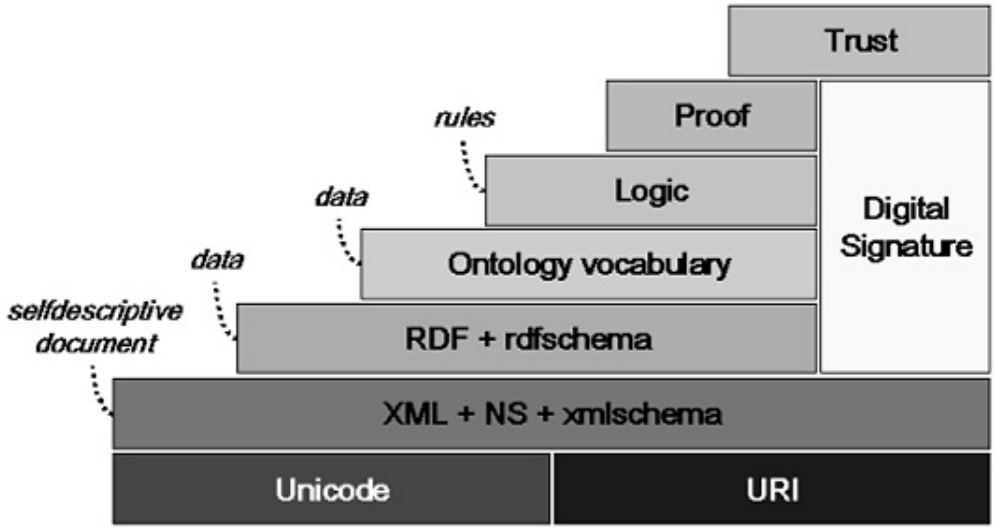
5.2 RDF and then...

Now that we have structured semantic information, the idea is to use several technologies to process this data, and hopefully lead us to more relevant information and navigation.There are different layers which form the Semantic Web as seen in fig. 5.4

- The Unicode and the URI layers ensure the use of international character sets and means of identification.

- The first layer of the Semantic Web is basically the RDF statemets

Figure 5.4: Layer architecture of the Semantic Web. (Koivunen and Miller, 2002)



- The next layer, containing XML, NameSpaces [1] and xmlschema [2], ensures that the semantic web can be integrated with other xml standards.
- The RDF + rdfschema layer ensures the ability to create the statements about subjects and objects by the use of URI's as seen above, as well as defining vocabularies that can be referred to by URI's
- The next layer is the ontology layer, which expresses the relationship between different types of things
- The Digital signatures layer provides means of keeping track of alterations to documents.
- After this comes the logic layer, which is thought to allow making inferences by the use of logic agents, it is still under research.
- The trust and proof layers are also important parts, though not quite developed yet. They are thought to provide proofs of authenticity and the means to give application means to determine wether these proofs are to be trusted.

[1] XML's way of uniquely identifying several XML languages used within the same XML language. To provide unique identifier strings, an XML namespace uses Unfified Resource Identifiers.

[2] An XML Schema is equivalent to a Document Type Definition. The difference being that it is defined in terms of XML itself, and has the possibility of contraining by datatypes such as strings, integers etc.

All these layers will play its role in the processing of RDF statements. In (Berners-Lee et al., 2001), Berners-Lee describes possible uses for the semantic web technologies. Let us say that you wanted to have your computer make an appointment at the doctors for you. Then you would ask your computer this question, and the computer would start a search. Through RDF statements, the computer could figure out where you live, when you have the time between work and other appointments to go to the doctor. Then it would find a list of doctors that specialize on what you need, where they are situated, and when they might have time for an appointment. Then your computer would use the logical layer to figure out a logically sound combination that might fit your schedule, and use the trust layer to verify that the information it is using as basis for the deduction is trustworthy. In the end it would return some suggestions that you could acknowledge or not, and then do the appointment. The semantic web is not yet fully operational, it is still somewhat out there in the future, but more and more pieces are getting ready for it.

5.3 What does RDF solve?

The RDF approach does offer a solution to our needs for richer metadata, especially since it gives us a standardized way of creating metadata within a more fixed vocabulary than XML in itself offers. RDF is already here, and has been used for many useful things already, such as feeds etc., and in the context of hypertext it has several constructs which could enable us to create more advanced hypertextual functions, such as typed, even directed associations. Also, RDF being what one could call an "identity-based" technology, it also brings us further towards a subject centered approach. But being mainly an envisioned part of the machine processable Semantic Web it is not in itself applicable as an immediate solution to information overload in the context of the already existing hypertext. It might though be used to enrich user experience with existing resources by marshaling the little metadata already existent in these resources, being somewhat of a top-down approach, but it will still not be able to do it persistently which could have solved many of our problems. RDF was not intended for top-down approaches to describing resources, its strength being to enable authors to add machine-processable information to existing network-retrievable resources, meaning we still have to produce a lot of metadata before the Semantic Web is going to reach its full potential.

Chapter 6

Subject-based classification: Controlled vocabularies or Ontologies?

In section 3, I identified the weakness of our current meta data classification of hypertext documents on the World Wide Web and suggested that we need to be able to say more about what resources are about than we can through the HTML <META> element. This is important to the findability of resources both by navigating hypertexts, as well as by doing searches. In (Garshol, 2004), Garshol identifies the "subject" property of the Dublin Core set of metadata properties as being the most useful property, since it is the only thing that explicitly describes what a document is about. What the "subject" property does, is to describe what resources are about by using *subject-based classification*. This can be done in many ways, and is generally combined with other techniques in order to create a complete solution. The following describes two different approaches to describing subject-based classification.

6.1 Controlled vocabularies

A controlled vocabulary is defined in (Garshol, 2004) as a closed list of named subjects which can be used for classification, also known as an "indexing language" in library science. The purpose of having a controlled vocabulary is to avoid authors defining terms which will result in the vocabulary becoming compromised, by for instance misspelling terms or chossing slightly different forms of terms leading to ambiguities. Examples of controlled vocabularies are for instance taxonomies and thesauri. Taxonomies are subject-based classifications that arrange terms into hierarchies, where realted terms are grouped together and cat-

egorized in ways that make it easier to find the correct term on is looking for. In (Garshol, 2004) points out that taxonomies are not themselves metadata, merely a way of arranging metadata. Thesauri are basically taxonomies to, allowing for subjects to be arranged hierarchically, but also allowing further statements to be made about subje cts.

6.2 Ontologies

Ontologies on the other hand, have open vocabularies resulting in much more descriptive power, due to the author being allowed to define the vocabulary at will. Ontologies, often referred to as Domain Models, are generally defined as a "representation of a shared conceptualization of a particular domain." They provide a shared and common understanding of a domain that can be communicated across people and application systems, and interweave human understanding with machine processability. They have been developed in Artificial Intelligence to facilitate knowledge sharing and reuse. (Decker et al., 2000) Alternatively, an ontology is a "logical theory which gives an explicit partial account of a *conceptualization*, designed in order to be shared by more agents for various purposes. A conceptualization is a set of concepts and their relations to each other". An ontology differs from other data models in that it is as concerned with the relationships among entities as with the entities themselves, and in the fact that the semantics of these relationships are applied uniformly.

When creating ontologies, or Domain Models, one has to create a model of the domain of interest, ususally consisting of objects and relations representing the different parts of the ontology. An ontology then, includes

- Entities, concepts (things)
- The relationships between those entities
- The functions and processes involving those entitites
- Constraints on, and rules about those entitites.

What is considered an ontology ranges from a simple taxonomy with an open vocabulary to a vocabulary of machine useable knowledge as standardized terminology and upward to a conceptual model with more complex knowledge representation, culminating in the notion of an ontology as a logical domain theory. (Park and Hunting, 2003) Ontologies in the context of machine interpretable definitions of a domain has several uses, most interesting to our context of hypertext and a machine interpretable web is:

- **Sharing common understanding of a structure among software agents**
  - this way, if several web sites contain information on the same subjects and share the same ontology, a software agent is then able to extract information on a subject from several sources

- **Enabling reuse of domain knowledge**
  - this is useful to support the sharing and common understanding of structures since a prerequisite for that would be that everybody use the same ontologies for the same domains, but also for securing that there won't be several ontologies describing the same domain. Also, bigger ontologies consisting of several subdomains would be able to be aggregated on basis of several exisiting ontologies.

Building an ontology is similar to that used to build an object oriented conceptual model, or an entity relational database diagram. The primary difference being that the other models must be modelled according to the various purposes they must fulfill, whereas ontologies are general models of a domain and thus are expressed independently of the various purposes they are to be used for. (Park and Hunting, 2003) An example of an ontology when defining it hierarchically, is defining it as a list of class definitions and definitions for relations such as "subclass-of" and "superclass-of" relating the different classes to each other. Relations are often binary, but can also be unary or n-ary, depending on the domain being described. Individuals in the domain are then assigned as instances of these classes.

6.3 What do Ontologies solve?

A key advantage of ontologies, is that the facts contained within an ontology can both be viewed and used by humans and computers alike. Thus, they can be used for more computerized tasks such as driving reasoning on the future Semantic Web being the basis for the Logic and Proof layers fo the Semantic Web Layer Architecture, but also be used by humans through an appropriate interface to aid us in solving some of our hypertextual problems with our existing documents. Ontologies in our context of hypertext and the web, make it possible to express resources on the web as concepts and relationships between them, thus enabling us to create much more information-rich structures than what we are able to express with HTML. Ontologies then are one of the prerequisites for making a transition to a subject centered web making it possible to create models by grouping the existing resources into subjects, classes of subjects, instances of classes and relationships between subjects. Currently there are at least two approaches available

for expressing ontologies by the means of markup making the model machine processable.

Chapter 7

The Web Ontology Language, W3C's language for Ontologies

(Consortium, 2004b) and (Consortium, 2004a) describe OWL as going beyond the capabilities of XML and RDF in representing machine interpretable content on the WWW. It is used to express representations of terms and associations between them, defining and instantiating ontologies for the World Wide Web. OWL is a result of RDF Schema (RDFS)[1] being found to be too lightweight and limited for advanced representation and inference of ontologies. It was thus created by enriching RDFS with semantics taken from DAML [2] + OIL [3] web ontology language which was the predecessor of OWL. It is intended to be one of the cornerstones of W3C's Semantic Web, and to be the first level above RDF to formally describe the meaning of terms described in RDF.

The OWL vocabulary is intended to describe such things as terms, classes, associations between them, cardinality, equality, typing and characteristics of properties. OWL is divided into three increasingly expressive subsets, each designed for specfic tasks and users:

- OWL Lite, for users needing a classification hierarchy and simple constraints.
- OWL DL, for users needing maximum expressiveness while retaining computational completeness [4] and decidability [5]

[1] Addition to RDF making it possible to define classes and class hierarchies, which are not possible to express with RDF by itself.

[2] DARPA Agent Markup Language

[3] Ontology Inference Layer

[4] All conclusions are guaranteed to be computable

[5] All computations will finish in finite time

- OWL Full, for users maximum expressiveness and computational freedom of RDF, but with no computational guarantees.

Being a component of the Semantic Web activity, OWL inherits the distributive features of the intended Semantic Web, meaning that it allows for information to be gathered from distributed resources. This is allow ontologies to be related making it possible to explicitly include information from other ontologies. Since descriptions of resources are not confined to a single file or scope, it is possible to define the same term differently, or to extend it in other related ontologies. The possibility of contradictions coming from this feature of OWL has to be dealt with by whatever application which processes it, but there is a required syntax and formal semantics found in RDFS that has to be followed to ensure that an OWL ontology can be interpreted unambiguously by software agents. OWL depends on constructs defined in RDF, RDFS, and XML schema data types, and thus uses XML namespaces to uniquely identify them within an OWL document. The same approach is used to uniquely identify other ontology vocabularies within an OWL document, making it possible to create derived ontologies.

Since OWL is intended to be interpreted by computers it also has the possibility of adding logical properties in order to provide support for logic inference on basis of the ontology. One can add set properties such as transitivity, symetry, equivalence, union, intersection and complement among others to the ontology. Another interesting feature is the possibility of adding versioning in the ontology, making it possible to track ontological changes within a domain.

7.1 What does OWL solve?

OWL is the World Wide Web Consortiums approach to adding logic inference to the Semantic Web layer model, as well as being their approach to expressing ontologies. In our context of hypertext, it extends RDF's contribution of rich metadata by enabling us to express richer structuring and associations on top of them, thus strengthening subject centered structure on the web.

Chapter 8

Topic Maps

Topic Maps, or the ISO-13250 standard is a subject-centered data model for indexing and representing any resource, be it addressable on the Internet or not. That the indexing is subject-based, means that the index contains terms representing subjects being proxies for words, concepts, and resources found on the Internet, in a book, or actually any resource imaginable. Topic Maps are said to originate from the idea of representing knowledge structures as traditional “back-of-book-indexes” (Pepper, 2000), and thus has, besides the terms, associative structures for classifying terms as in taxonomies, as well as associations such as cross references resembling the "see" and "see also" found in thesauri. These references can be thought of as typed associations pointing to information which is related with some or other degree of relevancy to the referenced resource in question.

In (Pepper, 2000), Pepper holds that traditional indexing techniques while working for traditional media are inadequate for our modern situation with computers doing automated indexing and information being found in multiple and disparate media, often demanding indexes to cover vast pools of information. They suffer from problems arising from ambiguity caused by i.e subjects having synonymous and homonymous meanings. These problems have been felt by anyone using web search engines, using full text indexing as a data model, being one of the main reasons for web searches returning an infoglut of mainly irrelevant hits. He sees Topic Maps as providing an approach that marries the best from several worlds, including those of traditional indexing, library science and knowledge representation, with advanced techniques for linking and addressing.

In (Garshol, 2004), Garshol defines the difference between indexing techniques such as taxonomies and thesauri on one hand and ontologies on the other hand as being a matter of having a controlled vocabulary or not, where the latter has

an open vocabulary. Topic Maps are very much resemblant of indexes such as taxonomies and thesauri, but they are not restricted to represent hierarchical structures as with taxonomies and thesauri, they are also far more expressive in that the do not have a controlled vocabulary, and may thus be considered to be ontology-based. Topic Maps being thus ontology-based, allowing for defining meaningful associations between concepts, adds the needed dimension to indexing that worked well in i.e "back-of-book-indexes". As Pepper discusses in (Pepper, 2000), the Topic Map way of indexing resources is much better adapted to our modern needs by having additional semantics added through the addition of ontological typing and structuring, as well as having other constructs similar to the extended functionality for indexing found in i.e glossaries and thesauri. Another advantage of Topic Maps being ontology-based, is that it gives us the ability to create an ontology, or a map that lies outside, or on top of the information being indexed. This top-down approach enables us to define structures and associations between resources independently of associations expressed within the resources themselves. This means managing and creating the meaning of the resource rather than just the resource itself, enabling us to take a step back and focus on the proverbial forest instead of focusing on the trees themselves.(Garshol, 2002). As Garshol puts it, it results in "an information structure that breaks out of the hierarchical straightjacket that we have gotten used to squeezing our information into."

The Topic Map model then, is made up of 3 basic constructs:

- Topics
- Associations
- Occurrences

8.1 Topics

The topic is the core concept within a Topic Map, and is an abstract entity representing any subject in order for us to be able to say something meaningful about it and attach relationships between it and other topics. A topic might be any thing that it is possible to talk about, whether it is a resource found on the Internet or not. It maps readily to entities or concepts in general ontologies described on page 6.2 as well as to the concepts in indexing languages. As with indexing languages, it is not the concept per se that is represented within the index, it is the term representing the concept, or names as they are called in Topic Maps. A topic may have any number of names, something which is not allowed in taxonomies

and thesauri due to the ambiguities it would cause. Also duplicate names are allowed in Topic Maps, which also is avoided by traditional classification systems, this is according to (Garshol, 2004) not a problem with Topic Maps, as additional properties, such as types, occurrences, and associations generally will distinguish the topics anyway.

8.2 Scoping topics

When making an ontology that describes a certain domain of knowledge, it is inevitable that one will get into situations where an entity has different definitions or meanings within different contexts. One would then want to be able to express the different perspectives within the ontology to ensure that the ontology is complete. This is possible in Topic Maps partly because it is possible to give any topic several names, and because any name can be given a *scope*. Scopes are topics that represent the different contexts or viewpoints one needs to express the different facets or perspectives of an entity, or to put it another way; one defines the contexts within which a certain topic name is valid. Scopes can also be applied to occurrences and associations, this gives the unique opportunity if one later needs to consult the ontology and wishes to filter the information on basis of the set of scoping topics defined within the Topic Map.

8.2.1 Typing topics

Everything in a Topic Map is a topic, and it is thus possible to use any topic within the Topic Map to type any topic making that topic an instance of the former. This allows us to group subjects into classes giving us a powerful structuring tool. Associations are also defined in terms of topics, and thus one can create any association type one would need, adding the much needed ability to create typed associations between resources.

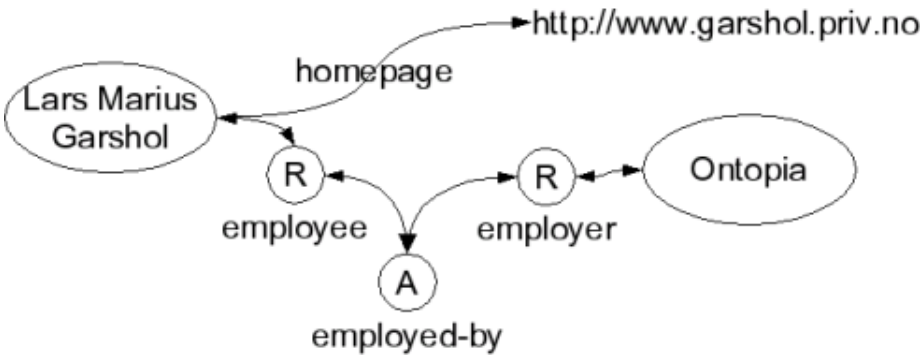
8.3 Associations

As mentioned above, associations are defined in terms of topics themselves and have the possibility of being typed. To do this one would create a template topic representing the association type, and give it a suitable name. Then one would create the associations themselves, defining them to be instances of the typing template one just created. The openness of this approach allows for any kind of relationship to be expressed.

8.3.1 Members playing roles

Topic Map associations are unique in that they have roles representing the involvement of each topic in the association, the relation thus going both ways. The structure of Topic Map associations are defined in terms of topics as members of a defined set of role players, playing a role within the association. Any number of roles may be played within an association, this gives Topic Map associations the flexibility to be defined as n-ary associations. The most common is for associations to be binary, i.e of the type "written-by" / "has-written" when referring to topics playing the roles of author and thesis. But they may as well be unary or n-ary, making it possible to represent for instance family structures with "child-of" ,"mother-of" and "father-of" roles. In (Garshol, 2003), Garshol's example as seen in fig. 8.1 shows how the nature of a binary association is built. Saying that Lars Marius "Garshol is *employed* by Ontopia" is the same statement as saying that "Ontopia *employs* Lars Marius Garshol", thus the direction of associations is not an issue with Topic Maps.

Figure 8.1: Binary Topic Map Association (Garshol, 2003)



8.4 Occurrences

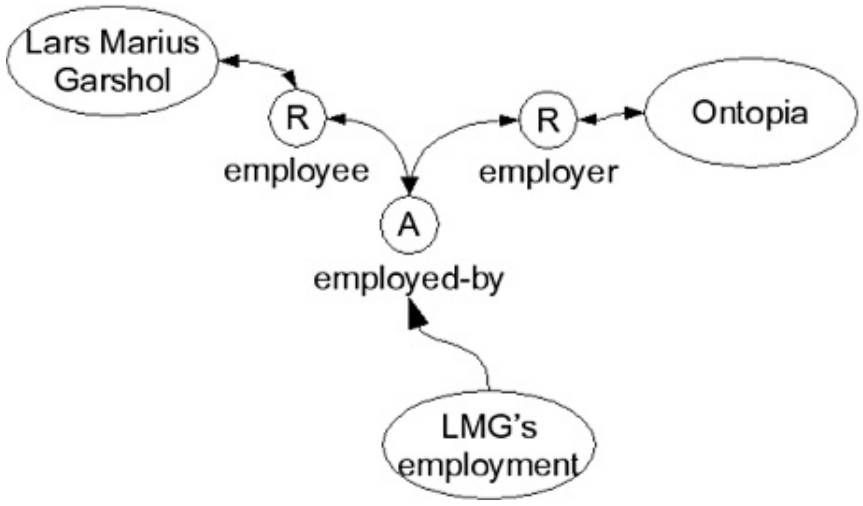
Occurrences are the resources themselves that are being mapped and pointed to. An occurrence can be anything that can be referenced to either on the Internet or in the physical world. It might be i.e a book or an article found in the real world, but usually, it is digital resources, found on a computer or on the Internet, be it textual or binary, a document , a sound or a picture. There are two types of occurrences, the ones that can be pointed to with some reference outside of the ontology itself, and internal occurrences, which usually will be a piece of information, i.e a

short describing text being a property of the topic having the occurrence. Occurrences may also be typed, making it possible to group occurrences as instances of a specified typing topic making them easier to filter and group.

8.5 Reification

Imagine that one has created some Topic Map construct such as an association or an occurrence, and one wanted to treat it as a subject in itself, making it possible to make new assignments from it as seen in fig.8.2 That is what we call reificiation. Or as Garshol puts it in (Garshol, 2003), it is creating a symbol that represents an assertion. Or even better; "thingification", since reification comes from the Latin word for **thing**, *res*.

Figure 8.2: Reifying an association (Garshol, 2003)



8.6 Identity

When creating Topic Maps ontologies it is of great use to be able to identify uniquely what entity one is referring to. Topic Maps have a fundamental distinction between subjects that are addressable, or digitally retrievable, and subjects which are non-addressable. In the case that the resource represented by a topic is network-retrievable, and that the URI can be used to identify topic uniquely, the address of this resource is called a *subject address*. In the case of non-addressable

resources, the URI can be used to identify the topic indirectly, and is called a *subject identifier*. The URI then references some information resource that provides an indication of the subject of the topic, a human could interpret this information resource and know what subject is being referred to. When an information resource is used this way it called a *subject indicator*.(Pepper and Schwab, 2003)

8.6.1 Published Subject Indicators

If a subject is likely to be represented in more than one Topic Map, ambiguity may occur unless one has a way of uniquely identifying the topics themselves across Topic Maps. Subject addresses and subject identifiers go a long way in identifying subjects, but are too simple to be sustainable in a broader context than the controlled environment of i.e a single Topic Map, or Topic Maps on an intranet. (Pepper, 2004) For this, Topic Maps have a construct called *Published Subject Indicators*, which are unique indicators to what subject one is referring to within the ontology. They are called *Published* Subject Indicators because they are published and maintained at an advertised address for the purpose of facilitating knowledge interchange and mergeability (Pepper, 2000), as well as being meant to be authoritative, stable and reliable as opposed to URI's. The PSI's follow the same scheme as seen above for Subject Indicators, where a *Published Subject* have subject indicators, namely *Published Subject Indicators*, where its subject identifier is the *Published Subject Identifier*. They have a similar function to XML name spaces in the sense that they will provide unique identification to topics when there is a possibility of inconsistency within the ontology due to ambiguity caused by for instance, topics having the same name, which might very well happen when merging Topic Maps as discussed below.

8.7 Merging Topic Maps

PSIs play an important role when one wants to merge two or several Topic Maps into one. Merging Topic Maps, means taking the union of the Topic Maps merged, and making a new Topic Map containing the result. This means that any two topics that meet the requirements for merging, will merge into one, having the union of all names, scopes, associations and occurrences attached to the new resulting topic. Since topics May have the same name, it is then important that there is a way of discerning if the topics in question are actually referring to the same subject. In early revisions of the ISO model, merging was to happen among other things also when topics had the same name, that would often result in the merging of topics, which actually were referring to different subjects. To borrow from the now well-known example from Italian Opera in (Pepper, 2000), there is

a distinct difference between Tosca the Opera and Tosca the cake, and merging them will create inconsistency as associations and occurrences for the cake will merge with associations and occurrences for the opera. It then makes much more sense to merge on basis of topics referencing the same PSI, ensuring data integrity.

8.8 Topic Map notations

The ISO-13250 Topic Map standard is an abstract data model, and is thus not constricted to be expressed by any specific notation. There are several alternatives to choose from, but the most notable is the XML notation for the exchange of Topic Maps called XML Topic Maps or XTM for short. Using XML as the exchange notation ensures standardization and the possibility of using several other XML technologies, such as XSLT-transformations, or XML Web Services for direct exchange with no need for a transformation from XML to some other format. Of course, since Topic Maps is an abstract data model, one would use a data base back end solution when possible, due to Topic Maps often becoming very extensive, and XML's wordiness adding to the workload. But for smaller projects, or in instances where one does not have access to a data base back end, the XML notation will suffice.

8.9 What do Topic Maps solve?

Topic Maps' focus on information retrieval give several good approaches to how our hypertextual problems could be solved.

Modularity and subject centered indexing - If we look back at some of the advanced hypertextual functions identified in older systems discussed in 1.4, the first that comes to mind, is the possibility of implementing modularity by defining a topic for any informational resource. This is also seen as important in the Polyscopic Information Modeling approach as discussed in 2.4, which draws its insistence on the conscious creation of information in small manageable modules from the lesson thought us by the creation of high level approaches to programming such as object orientation. As soon as we have topics reifying an information resource, it is much easier to assign additional hypertextual functions to it.

Structuring of information - As several of the hypertext pioneers identified, there is need for structuring of information. Usually this structuring has been hierarchical, something which is possible to do with Topic Maps by using class / instance structures to create taxonomies. But it is also possible by clever use

of associations and scopes, to create other structures which are not necessarily hierarchical. An example of such structuring is discussed in 2.4, where the authors advocate for a polyscopic structuring of information, which could perfectly well be done with Topic Maps as well.

Filtering of information - Both NLS and Intermedia, known hypertext systems created before the World Wide Web supported constructs that allowed for filtering of information in some or other way. Also the authors behind the polyscopic approach see filtering by perspective or context as being one of the most important navigational and information modeling aids. This is of course possible by Topic Map scoping, making it possible to scope names, associations and occurrences. But also through typing which effectively groups information in to classes an instances, becoming useful navigation and filtering aids as well.

Extended link functionality - Both Randall Trigg's Textnet, Englebart's NLS, Browns early Fress, the later Intermedia,and Nelson saw the importance of typing links to diminish the problem of getting lost in hyperspace. Topic Maps pose powerful tools for implementing link typing due to the architecture of associations, being made up of roles played by topics. Both the name of the association topic instance itself, as well as the names of the topics representing the roles, and also the names of the topics playing the roles are available for use. These names could be used to add information to the starting point of an association, telling the user about the nature of the association, what relation the starting point has to the target resource etc. giving the user much better clues to what the association is about, and to what kind of context she is taken to. Also, there are extended possibilities in reification of associations, which would given even further possibility to add metadata to associations. Another possibility becoming possible since a Topic Map can be seen to be an external link base, is to fetch any metadata available for the target topic and display it for the user, giving her a further peek into what the target resource is about.

Metadata and structuring - One of the main features of Topic Maps, is the ability to define both metadata and ontological structuring with impressively simple means. For creating metadata, its subject centered approach gives us good means of representing information resources as subjects, namely topics, and then to express for instance Dublin Core metadata properties by mapping names, occurrences or associations to them. Through reification it is in addition possible to attach metadata to not only the resource themselves, but also to structuring elements found in the ontology as mentioned above.

High Level Info - The authors of (Guescini et al., 2005) argues that much of our information overload problems can be attributed to the fact that there is too little high level information found on the web. Meaning that users are presented with too many details without any high level information that could help us make sense of what the detailed information is really about. This is also seen as important in old systems such as "Hyperties" and Brown's "Intermedia" discussed in 1.4.6. Topic Maps are in them selves a top-down approach by describing existing information resources, residing outside the resources. The effect being that when a topic reifies an actual information resource, one is able to describe that information resource, giving a high level view, as well as the possibility of relating several high level views in structures, creating a coherent system of high level views as a navigational and structuring metaphor as proposed in 2.4

Vannevar Bush had a vision and saw it in light of the possibilities posed by the birth of new technology in his time. This inspired many of the later hypertext pioneers into driving the vision and the research further by the means of the technologies of their time. Topic Maps is part of the newly emerged technology of our time, and such is a possible solution to implementing the insights reached by the pioneers of the hypertext fields in unison, as well as having the possibility to be instrumental in the production of advanced hypertext on the World Wide Web.

Chapter 9

RDF or Topic Maps?

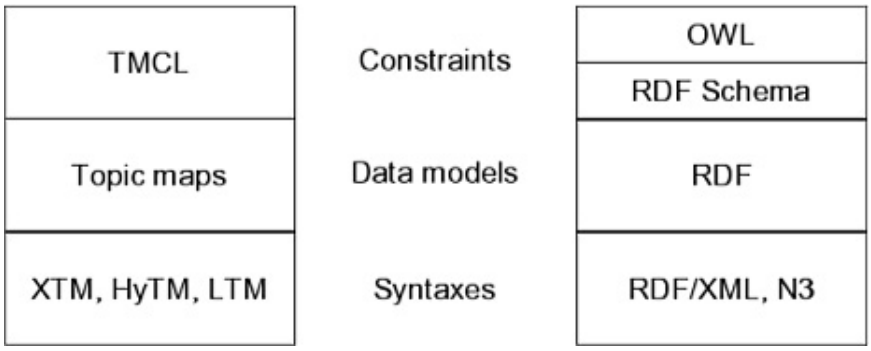
When looking at RDF + OWL and Topic Maps, one will see that they share many similar traits. One reason why there are two so similar standards available is that the need for meta data frameworks and ontology-based languages was apparent to many at some point in time, but there was no communication between the two groups creating the two frameworks. It was later, when the frameworks were made public that one saw how similar the two solutions were, but at that point there were too many issues, both technical and political, making it impossible to merge them into one standard. Also the two frameworks are intended for similar, but also somewhat different purposes. Topic Maps where created to support high level indexing of sets of information resources to make the information in them findable as well as giving a more high-level view of the subject domain. RDF on the other hand, was intended to support the vision of the Semantic Web through providing structured metadata about resources and a foundation for logical inferencing. The differences in outlook also make it harder to merge the two technologies into a single one, since the communities do not have the same goals.(Garshol, 2003)

9.1 Similarities

In (Garshol, 2003), Garshol presents the diagram in fig. 9.1 and identifies that both standards have three areas that coincide within which the various technologies or notations fall.

- Both standards families have several interchange syntaxes based on XML
- Both standards are abstract data models not dependent on any notation

Figure 9.1: The two standards families (Garshol, 2003)



- Both standards need additional constraint languages to express what is allowed within the data model.

Other similarities between the two standards

- Both standards are identity-based techonolgies, meaning that the key concepts are symbols representing identifiable things which statements can be made about
- Both standards have ways of indentifying nettwork-retrievable resources as well as subjects which are not.
- Both standards have the ability to define ontology-based structures, as well as more taxonomy-like structures.

9.2 Differences

- RDF is intended for situations in which information is to be processed by applications, while Topic Maps are intended for human consumption.
- There is only one way of making assertions about subjects in RDF by the use of *statements*, while the Topic Map model has three different *topic characteristics*, being *names*,*occurrences*, and *associations*
- In Topic Maps, associations are bi-directional, where topics play roles within the relation, also it is possible to have more than two roles in a relation. In RDF, having an association involve more than two members is not possible, also statements are directed having subject and object roles, in essence making the associations directed.

- Both standards have a way of identifying the identity of resources, but the Topic Map standard makes a distinction between a *subject address* where the subject identified is the resources itself, and a *subject identifier* where it is whatever is described by the resource. RDF does not have this distinction.

- Reification in Topic Maps uses the same constructs as for creating other Topic Map elements, while in RDF one has to create a special statement, making the assertion different from what one gets when creating a normal assertion. This makes it heavier to process reification in RDF du to the extra overhead needed

- Scoping subjects in Topic Maps is a built in feature of the data model itself while RDF does not have this feature at all due to RDF statements not having any identifiers which one can reference. It can be attained by reification though.

- Topic Maps are more high-level than RDF in the sense that a Topic Map contains more information about itself than does the RDF model. On the other hand, this makes the RDF model more light-weight than Topic Maps.

- Some, but not all of the features which RDF lacks in itself when comparing it to Topic Maps, can be attained by the use of OWL features. This would be the more descriptive features of OWL which adds extended semantics to the RDF resource triples.

Despite the many differences between RDF and Topic Maps, there is a growing consensus that the need for some way of interchanging information between the two standards is getting more and more important. The World Wide Web has recently formed a "RDF/Topic Maps Interoperability Task Force" to look into how information might be exhanged between the two approaches, being a starting point for establishing standard guidelines for RDF / Topic Map interoperability. In (Practices and Group, 2006) it is stated that the purposes of the to technologies first appeared to be very different, however, lately it has turned out that they do have a lot in common leading to calls for their unification. There are no signs of any movement towards merging the two approaches in order to reach one standard that can be used uniformily, it might not even be desirable. It might be that it is more desirable to have both technologies develop and mature themselves into perfection of what they do best, and then to reach a more wholistic approach by letting the two technologies collaborate when needed.

9.2.1 Why choose Topic Maps?

Looking at the similarities of the two standards tells us that in many cases, it would not really matter which one of the two standards one chooses, and in (Garshol, 2003), Garshol shows that it is in many cases possible to make mappings from RDF to Topic Maps and vice versa. Which of the two standards to choose will depend on the task at hand and the resources available. When choosing a standard for a system that is solely going to be processed by computers it seems natural to go with the RDF family, especially if the application is going to be dependent on extensive logic inferencing, where OWL has many capabilites that goes far beyond the Topic Map model in itself. With the future TMCL [1], it might be possible to do it with Topic Maps too, but as RDF + OWL was designed for such a task it would still be the natural choice for the task.

Looking at some of the differences between RDF and Topic Maps will show in which situations one might want to choose Topic Maps over RDF. The most notable differences being that Topic Map associations are multi-directional in that they may have more than two role players, also the lacking capability of natural scoping of subjects in RDF would be notable in situations where one needs to do many-faceted information representations as proposed in 2.4. More importantly in the context of knowledge presentation as an alternative to our current implementation of hypertext, Topic Maps being more of a high-level model than RDF, might make it more suitable for situations where the information isn't going to be processed solely by computers, and one needs a more human readable description of the subject domain. In (Practices and Group, 2006) Topic Maps are considered a technology for "knowledge integration", meaning that it is used to "...synthesize multiple knowledge models into a common model...focuses more on the synthesizing of understanding of the same subject from different perspectives..." (Wikipedia, 2006b), it is focused on modeling knowledge based on the knowledge itself making the knowledge available to humans, as opposed to the RDF approach of making data more accessable and more richly described in order that machines may understand it.

9.2.2 A possible hypertext implementation?

As I discussed in 8.9, Topic Maps pose several solutions to the implementation of many of the missing advanced hypertext functionalities envisioned by the early hypertext pioneers, and could thus be considered as a suitable model for how the representation of information could be done. Also in the light of Polyscopic Mod-

[1] Topic Map Constraint Language

eling discussed in 2.4, Topic Maps are seen as a good tool for implementing the Polyscopic Information Design methodology for designing the missing high level information by scope. The RDF / OWL aproach solves its part of our problems as well, such as the production of better metadata for machine consumption as well as advanced ontology-based inferencing, and is as such a great tool for solving our problems for the underlying need for tightly marked up data leading to better expressivity and decidability for computers.

A possible future solution for a proper implementation of hypertext on the web would probably be to use RDF/OWL and other Semantic Web technologies for the bottom-up approach, securing good metadata, allowing for good searches and machine processability, and use Topic Maps top-down for designing data and information into knowledge structures for human consumption. Being technologies that demand a more advanced framework than HTML over HTTP does, there would in addition to a change of underlying framework, be a need for the production of good tools for both perceiving and editing the hypertext, such as editable browsers, good abstractions for easy creation of ontologies and hypertexts based on the underlying technologies, NLS-like systems where data taken from several sources, web, email, newgroups etc are seen as just parts of hypertext to be marshalled by RDF / OWL and then presented by Topic Maps.

Chapter 10

TMemex

The framework is still not here for the web to make a swift transition from a document centered web to a subject centered one with tools created to make the most out of it. But it is getting there, and it may be that with the new technologies the possibilities of creating Bush's vision of the trail on the web itself could be a possibility. Doing such a thing would either require some kind of external link database as seen in several of the pioneering hypertext systems discussed in 1.4, holding all the information needed to create the trail, as well as holding additional data such as annotations, versioning, etc. Or the fundamental base of how we interact with documents on the web would have to change completely into something closer to what Berners-Lee intended in the first place, a collaborative environment with browsers that allow for editing and collaboration as well as being instrumental in creating correct, meta information-rich content. As the new ways of thinking about information on the web takes hold, and more and more well defined content is being produced by RDF / Topic Maps-aware authors and applications, it might be that we will reach such a fundamental change, but it will happen gradually. In the meantime, alternative solutions will flourish and continue to contribute to the advancement of hypertext, though resulting in the usual plethora of alternative solutions existing in lack of a standardized solution.

10.1 Metadata for me?

Currently, it seems as though current envisioned or implemented solutions group themselves into either becoming a function of current browsers, or projects trying to implement an Open Hypertext System model. An OHS makes it possible to express links in a manner independent of the storage location, known as external link bases. An example of a browser extension would be "Piggy Bank". Piggy Bank is a part of W3C / MIT's joint "Simile" project, seeking to enhance col-

laboration among digital assets, schemata/vocabularies/ontolgies, metadata and services. "Piggy Bank" is a browser plugin based on RDF, that lets users make use of Semantic Web content within web content as users browse the web. Where Semantic Web content is not available, Piggy Bank can invoke something called a "screenscraper" to restructure information within existing web pages into Semantic Web format (Huynh et al., 2005). This project actually implements both approaches, as it also has created "Semantic Bank", a web server application that lets Piggy Bank users share the Semantic Web information they have collected, enabling collaboration, creating a sort of external metadata base. Another example of creating an external link base is described in (Cianciarini et al., 2002), giving the users the chance to build private, dynamic, multi-destinational, multi-directional links in external link databases by the use of XLink and XPointer.

Looking back at Bush's Memex and its trails, one could say that the approaches described above are possible versions of trail making, at the same time as making grounds for sharing and collaboration. The focus of both these approaches is the creation and sharing of personal links and metadata, triggered by both the problems of the weakness of our existing hyperlink scheme, as well as what (Huynh et al., 2005) calls a "chicken-and-egg" problem, that the Semantic Web proposes a solution to our current problem, but to do so it needs more Semantic Web content to be created. We are in some ways in a situation similar to what Bush described, we have information overload problems, the solution is thought to lie within the birth of new technology, so we envision and attempt new ways of structuring information hoping that it may help us in our needs for properly designed information, and that it may further inspire, inform other people to keep the vision going.

For the practical part of this thesis, I am going to use the Topic Map standard to create a Memex-like system using Bush's notion of the trail for personal metadata. The main reasons for choosing the Topic Map standard is to test whether it is possible to use it to create the trail-function of the Memex. The reasons why it is interesting to see if Topic Maps is a suitable tool for the task, is its inherent subject-oriented architecture for making the nodes in the trail as well as the possibility of typing the nodes, the typed bi-directional associations for creating meaningful, annotated two-way relations between the nodes, the scoping ability for filtering and for adding context to side branches, the merging for the ability of implementing the merging of trails being based on the notions of PSI's assuring that any trails being merged will do so correctly. Another reason for choosing Topic Maps is its top-down approach to structuring information as discussed in chapter 9, and how that maps closer to how Bush's trails enabled the user to construct her own categorization structure on top of existing resources, creating an

interaction layer supporting a better way to interact with the underlying resources, closer to how the mind works.(Schraefel et al., 2005)

10.2 Implementing the trail

Bush's microfilm database is of course replaced by digital media, mainly web pages, but any digital resource is of course possible as long as it has some form of URI scheme that makes it addressable. Using Topic Maps as the basis for the trail also gives the possibility of representing more abstract resources within the trail by creating nodes to represent subjects which may or may not point to a PSI identifying the subject uniquely. One could then imagine improving the trail to be able to add concepts per se without them pointing to any resource, but it makes the most sense when the abstract subject has an addressable resource as an occurrence in the case of this application since it is based on resource existing on either the web, or on the users computer.

Secondly, the application attempts to implement on one side Bush's notion of the Memex having two screens where documents or other resources could be juxtaposed in order for the user to able to compare resources that are to be associated. On the other side, it is also an attempt to implement Nelson's parallel textfaces, specifically his notion of removing the initial structures imposed on the information in order to be able to create one's own structure to be applied on the information. This is an attempt to address the problem of loss of context when linking to information found in the midst of an other document as discussed in 2.1.1

Nelson's idea of stripping a document of any existing information, and then to create indexes pointing to the words or other elements within a document, makes it possible in theory to target any word or collection of words or elements within a document. It is then possible to create a system where one assigns subject centered metadata to pieces or collections of data found within a web document, instead of having to attach meta information solely to the document as a whole, improving on metadata granularity. This way it is thought that one can decrease the degree of loss of context that users might experience when following the normal untyped links found within HTML hypertexts as discussed in the introduction. There is however a downside to this approach, since basing the indexes on the position of words within a document is very fragile approach due to the changing nature of documents on the web. As soon as a document is altered and content is added or removed, the indexes will no longer point to the intended content. With Nelson's versioning of documents this would not have been a problem at all, since new

versions would not cause the original resource to be altered, but for our current situation a more robust solution might be needed.

10.2.1 A useful addition to the trail?

One thing that Bush did not think of when envisioning his trail, was the possibility of adding contextual information to nodes and branches. Implementing the trail with the use of Topic Maps, adds the possibility of doing so. Our implementation will not do so, but one could imagine the usefulness of adding scoping to nodes or branches, creating the possibility of filtering the output of the trail, something which probably would be helpful in cases where for instance the trail becomes very large.

10.3 Technical Solution

To make a personal metadata system, it would have to be something that happens on the client side, where information could be saved locally in order to access it quickly and safely. The solution could be rendered as any desktop application, but as the web browser has become the main interface for interacting with web content, it might be a good idea to either create a browser plugin as we have seen with "Piggy Bank" (Huynh et al., 2005) or "XlinkProxy" (Cianciarini et al., 2002), or even a full fledged browser since we also imagine the browser having parallel document windows.

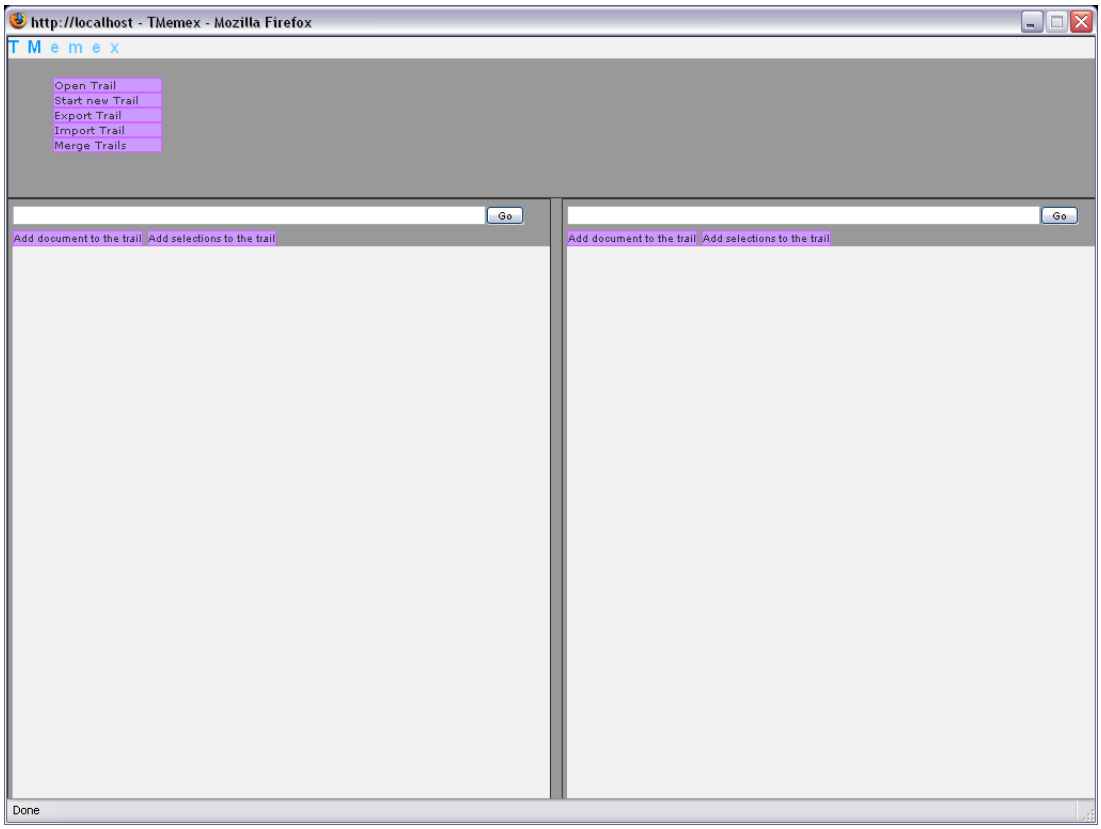
10.3.1 The thought browser

The optimal solution if a browser of the kind was to be produced, would have been to have it implemented in a low-level programming language such as C or C++, making it as fast and efficient as possible. The trail Topic Map itself would probably be best implemented through the use of external files in the form of XML Topic Maps, since they need to have the possibility to be created from scratch and to be constantly updated in order to let the trail grow, and last but not least; to be exported, shared and merged.

The browser is thought to be divided in three parts, the two horizontally paralleled windows with an address field above each of them for entering URI's to enable the fetching of resources. As well as buttons or the like to signal that one wants to add a certain resource or part of a resource to the trail. The buttons could be likened to the blank code spaces and pointers that Bush envisioned in his Memex. In addition to these buttons, there could be buttons for additional

hypertextual functions to be carried out. The third part is thought as an area of the browser that allows for manipulation and visualization of the trail itself. A thought browser is depicted in 10.1

Figure 10.1: The thought arrangement of windows in the browser



To allow for manipulation of the trail, a dialogue between the user and the software would be required. The user would have to enter the information that is to be assigned to the elements of the trail in order to create the semantically rich nodes and associations that are needed. One could imagine using pop up dialogues asking questions and giving the possibility of entering the information needed, at least for situations where a simple input is required. There will though, be other situations where several items of information are to be entered as well as having to give the user a choice between several alternatives when entering the information, and doing this with pop up dialogues could be cumbersome and

confusing for the user. In these situations it is probably more appropriate to use the trail area for these purposes since it is already defined as a part of the browser, and also would give a better overview when entering several pieces of information that have a relationship to each other. The trail area should have enough space to make that possible, as well as making it possible to display or visualize the trail itself in order to navigate it.

Visualization and navigation of the trail could be carried out in several ways. Bush himself envisioned the user tapping keys and entering the code for a specific trail which would bring up the head of the trail, being the first node, and then to sequentially navigate the trail in the order nodes were inserted, almost like reading a book. That would maybe make sense when creating a guided tour where the sequence of the node is part of the intention of the trail, but if one wants to give the user the possibility of navigating the trail as a hypertext, where she is free to choose an her own path through the trail, the user must be presented with a high level overview of the trail. How the actual rendering of the trail is done should be a matter of letting the user choose her preferred visualization, alternating between textual presentation by the use of HTML to render hyperlinks and textual descriptions for nodes and associations between them, or a more visual approach by visualizing graphical clickable nodes and arcs and their descriptions.

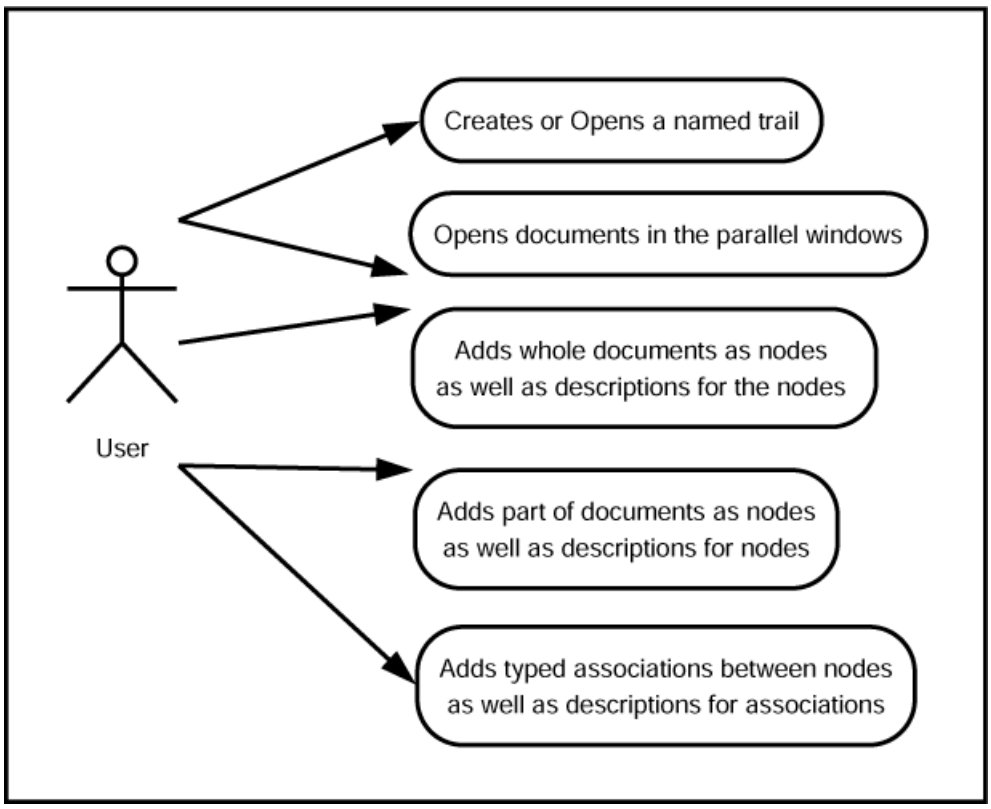
10.3.2 Use case: populating the trail

A thought use case of the user creating and annotating a trail is depicted in figure 10.2. To make a parallel to Bush's descriptions of his visions for the Memex, one would say that the user by creating or opening a named trail "...is building a trail, he names it, inserts the name in his code book, and taps it out on his keyboard...". Next, the user opens documents in the parallel windows by entering the URI in the address fields adjoining the document windows and thus follows Bush in "...if the user wishes to consult a certain book, he taps its code on the keyboard, and the title page of the book promptly appears before him, projected onto one of his viewing positions...". The difference of course being that Bush envisioned books having been photographed onto microfilm, and thus also having levers to flip through the book sequentially, while the browser mainly will open existing hypertext documents or other addressable resources.

The next point in the use case is the user adding whole documents or parts of documents as nodes, as well as descriptions or annotations for these nodes. Bush describes it as "...before him are the two items to be joined, projected onto adjacent viewing positions...the user taps a single key, and the items are permanently

joined...". Bush also mentions marginal notes on the documents, which probably could be possible if one adds descriptions annotating the nodes. These descriptions or annotations could then be recalled when viewing the trail and be displayed together with the resource in question. When Bush says that the items are permanently joined he envisions it as the Memex attaching dots for photo cell viewing to the documents in questions, holding the index number of the other item in the Memex. This would in our case be saved in the Topic Map. This is implemented by the next step in the use case: the user adding typed associations with descriptions for the associations as well. As Bush puts it: "....associative indexing, the basic idea of which is a provision whereby any item may be caused at will to select immediately and automatically, another. This is the essential feature of the memex...". Bush does not specify how the associative indexing is to be done, but by the use of Topic Maps for creating the associations we have added the power to type and annotate the associations as well as associations being bidirectional.

Figure 10.2: Use case of user populating a trail



10.3.3 Use case: Using the trail

As the trails are named, it is easy for the user to open a trail of interest and have it rendered in her chosen format in the browser's trail area. Bush envisioned that "...A touch brings up the code book. Tapping a few keys projects the *head* (emphasis mine) or the trail...when one of these items is in view, the other can be instantly recalled by tapping a button below the corresponding code space. Morevover when numerous items have been thus joined together to form a trail, they can be reviewed in turn, rapidly or slowly, by deflecting a lever like that used for turning pages in a book...".

This is where this implementation differs from Bush's idea of starting with the "head" of the trail and then navigating it sequentially, in that the whole trail is presented, giving the possibility to click any node within the trail, and to follow whatever association the user is interested in following. In the case that the trail is represented textually which may not give as good an overview as a more graphical visualization might give, one has to ensure that the user knows where she is within the trail. The trail should then support some form of "bread crumbs" (Nielsen, 1999) or contextual clues, ensuring that the user does not loose her context within the trail. After clicking a node, the browser displays the given resource in one of the adjacent windows after having asked for an indication of which window she wants to use for display. If the resource represented by the node is a whole document, the document is fetched and displayed as it is, if on the other hand the node represents information found within the resource, the resource should be displayed and the parts highlighted as well as scrolling the window to the position of the highlighted content. Also, the browser should retrieve any annotations attached to the node and display them.

Bush does not mention the possibility of altering the trail, but this could of course be possible. One could imagine the possibility of altering the individual nodes and arcs, or even, creating new versions of the trail, retaining all versions, marking them up with versioning information. One thing that Bush finds of great importance though is the sharing of information, and the next points in the use case depicted in fig. 10.3 are all about that. Bush puts it this way: "...It is an interesting trail....so he sets a reproducer in action, photographs the whole trail out, and passes it to his friend for insertion into his own memex..." This translates directly to the points in the use case where the user exports the trail and the point where the user puts an external trail into her browser.

The next point in the use case, where the user merges trails, is not directly described in Bush's vision, but surely it is a function that might lead to fulfillment

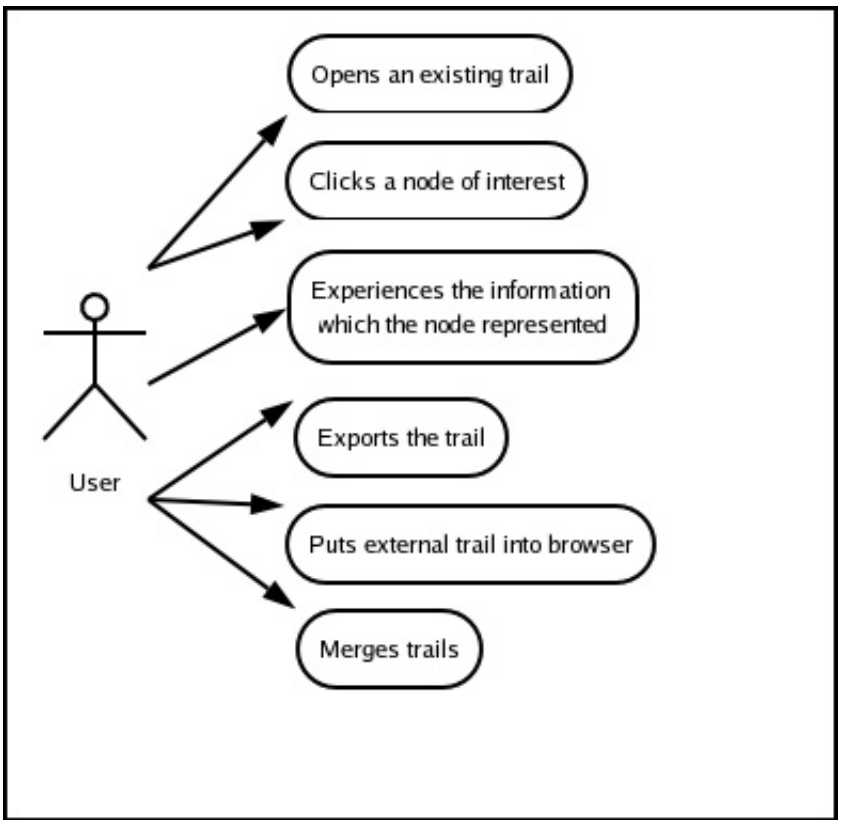
of Bush's wish for a system that allows for collaboration between people from different professions. In writing "...photographs the trail out, and passes it to his friend for insertion into his own Memex, there to be linked into the more general trail...", Bush foresees the possibility of connecting trails in some way in order to bridge information coming from different disciplines, which he sees as important to combat the problem that interdisciplinary research is loosing out on important information due to specialization and fragmentation.

Bush sees the solution to this problem as connecting two trails by association, making one trail a side trail a branch of another. Merging on the other hand, when done with Topic Maps and its ability to merge resources based on subject identity, adds an extra dimension to the trail by expanding the subject nodes themselves consisting of the union of all associations and annotations for each node being merged together. This possibility gives a whole other overview of the information from the different disciplines than having them organized into a hypertextual network. All information about the same subject is grouped together and all associations about the same subject run from the same node, instead of having information about the same subject spread in a hypertext network where one has to navigate one self around to find pieces of information that share the same subject. A side effect of merging information from different disciplines based on subject identity is the possibility of new knowledge being revealed as a result of combining several resources or associations for the same subject, which might have passed unnoticed if they had not been grouped together.

10.4 Implementation of the browser

The implementation itself is created by the use of different technologies than a live browser should be using, and is made to give an example of some of the functionality the real "TMemex" browser could have. Since the author of this thesis already had implemented a Topic Map parsing engine in the server scripting language PHP Hypertext Preprocessor, it was a natural choice to reuse some of its capabilities for the implementation of the browser. This made it possible to both create new trails based on Topic Maps as well as to edit them with ease, but also to extract them easily for presentation. Since the implementation was already using PHP as a scripting language, and since the author is trying to create a system that has the ability to manipulate selected parts of the retrieved documents on basis of Ted Nelson's parallel textface idea, it requires some degree of string manipulation which is performed with great efficiency by PHP.

Figure 10.3: Use case of user using an existing trail

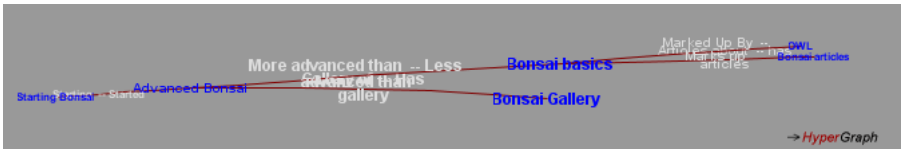


The browser interface and its client side functions itself was built using the Mozilla Firefox web browser, HTML, Cascading Style Sheets, and the abilities found in the Document Object Model for HTML when accessing it with JavaScript. The result of using all these together is also known as Dynamic HTML, and is needed both to render the browser itself, as well as to manipulate the information found in the documents, and to send them to the server for processing and addition to the trail. One normally has to use either the GET or POST methods of the HTTP protocol to send information from the client to the server, and doing so results in a state change where the URI of the browser changes to refer to some other file referred to by the GET or POST request. The result of doing so would cause the browser holding the rendering of the implementation to not only change state, but also to loose any values held in memory by the referring web page. To prevent that from happening, the implementation makes use

of AJAX [1], a technique that uses the XMLHttpRequest object of the Document Object Model to exchange data asynchronously with the server. This makes it possible to do GET and POST operations in the background while retaining the same document URI within the browser. It is important to point out that using AJAX for anything that is supposed to represent a real solution to hypertext problems on the web would require an extensive discussion, as it has several usability issues in that context. But the use of AJAX in this thesis is just instrumental to implement a thought browser, and is thus out of that context.

The trail part of the browser uses a graphical interface to better illustrate the possibilities of how a trail could be visualized and navigated from a high level point of view. Even with good structuring and metadata, one might experience information overload when the amount of information gets very large. It is therefore important to find good metaphors for visualization of the trail that diminishes the sense of information overload, and a textual interface may offer less possibilities of doing that. With Topic Maps one has of course the possibility of using scoping to give a way of focusing on smaller parts of the total amount of available information at one time, but as this version will keep as close to the original idea of the trail as possible this is not implemented. An approach that manages to give the user a possibility to focus on parts of the trail without losing the overview is implemented in the open source "Touch Graph" Java applet. It visualizes nodes and arcs with their correspondent names, giving the possibility to click on an area of the trail to bring it into the center of vision as seen in fig. 10.4, while other parts of the trail is sent to the back thus not overcrowding the current vision. It also gives the possibility of assigning URIs to the nodes and making them clickable in order to open the URI represented by the nodes in a web browser, something which is needed to provide the possibility of surfing the trail. The applet uses its own XML language to represent nodes and arcs, so the application will on opening a chosen trail export the Topic Map into this notation and then pass it to the applet in order to have it visualized.

Figure 10.4: The Touch Graph applet visualizing a trail



[1] Asynchronous JavaScript and XML

10.4.1 The parallel document windows

The two adjacent screens envisioned by both Bush and Nelson to help associative thinking could have been implemented in several ways in HTML. Using a frameset or the <iframe> tag, could be seen as a easy way to retrieve documents as just passing the URI would be enough to retrieve the resource referenced by the address entered by the user. But the extra complexity needed to target resources in the frames and then from the frames an back again made it unnecessarily complicated. Also, since I needed to process the resource identified by the URI before displaying it, this would have made it necessary to write the result of the PHP parsing to a file, and then to use extra JavaScript to have the frame open the resulting file. Thus it made much more sense to create the parallel document windows using two elements, each of them having unique ids. This way, it was just a matter of addressing the div element's **innerHtml** property and assigning it with the result of the computation being returned from the AJAX call to PHP. Then by using CSS' "overflow" property for the <DIV> element, it was possible to render scrollbars whenever the vertical length of the resource retrieved made it appropriate.

10.4.2 The Topic Maps engine

The engine uses a relational database as a back end storage, and is thus dependent on having access to some sort of relational database connection, something which would be possible for a live browser since it is supposed to run on the users computer but not desirable since it might demand to much of the client system. One could though, imagine the possibility of the browser having both possibilities, to provide for situations where trails get very big, or the resources on the client system are sufficient. The open source relation databases MySQL, or PostgreSQL are good choices for running the implementation. An SQL batch file for creating the database schema can be found in the appendix.

In addition, since the implementation uses PHP, it is also dependent on running on a web server, which a live browser is not meant to be dependent on since it is meant to work as an ordinary browser, fetching resources on the Internet or on the local files system. Currently Apache.org's Apache HTTP server, or Microsoft's IIS supports the use of PHP and are thus good choices for running the implementation.

The engine itself is programmed with an object oriented approach where all Topic Map elements are implemented with a class of their own as well as having the appropriate getter and setter methods to facilitate population and extraction

of the topic map data in memory. Since this implementation uses a database back end, the author has also included PHP code which exports and translates the Topic Map data from an SQL schema to the object oriented data model and back again. The object oriented data model has only getter and setter methods for extracting the basic Topic Map elements, thus it is necessary to have additional functions which utilizes these methods in order to drill down into the data model to get at the information itself and to render it for the intended use. All the PHP code used in the project is included on the CD-ROM that comes as part of this thesis, while the code relevant to the main implementation is included in the appendix.

10.4.3 Building the trail

To build the trail, all the technologies mentioned have to be set in action. HTML and CSS is used to render the browser described in 10.3.1 including the user interaction interface consisting of buttons with JavaScript event handlers attached to them. These event listeners will call their assigned JavaScript functions. If taking the use case in fig. 10.2 into consideration, what happens when the user clicks the button assigned to enable the user to open or create trails, is that the assigned JavaScript function does a GET method AJAX call through the XMLHttpRequest object for a PHP script which renders the dialogue interface by the use of HTML forms and sends it back to the JavaScript functions which by the use of JavaScripts ***document*.getElementById(id).innerHTML** populates a waiting - element within the trail area. The user then fills out the desired information in the case of creating the trail, or chooses from a dropdown list of existing trails which PHP has fetched from the "topicmaps" table in the database back end.

Instead of submit buttons which would have caused the browser to change its state, a dummy button having an event handler attached is clicked making a new JavaScript function call. The function extracts the data found in the HTML form elements, builds a URL encoded string on basis of the extracted information, and sends it by doing a new AJAX call to the server, where the PHP script parses the URL encoded string and inserts the data entered by the user into the database. The trail's unique id is sent back as the PHP script's response to the call, and is kept by the browser as an argument to future edits on the chosen trail.

Next, the user opens documents in the browser's document windows by entering URIs in the address fields, and chooses which of the parallel windows she wants to display the chosen resource in, resulting in the resource being displayed. Now the user has the possibility to add this resource to the trail by interacting with the user interface. The two possibilities, of either adding whole documents

Figure 10.5: Screen shot of add document dialogue

or adding parts of documents require different actions by the software. When the user chooses to add a whole document, a new dialogue is presented to the user following the description above with the difference of having the chosen resource's URI sent as an argument to the JavaScript functions. The user is asked to enter names for the new nodes as well as a description of the node. If there are any existing nodes already present in the topic map, the user is presented with the possibility of assigning associations between the new and existing nodes, thus building the trail. A screen shot of the dialogue can be seen in fig. 10.5. In order to give the user the possibility of creating her own association types, she may create names for the two roles being played in any of the associations within the trail by presenting the user with text fields labeled "Enter name for the role played by the new node" and "Enter name for the role played by old node". This limits associations to being binary, but one could of course imagine the possibility to create n-ary associations in a real browser. Following the description above, AJAX calls are made to PHP on the server side, which will extract the information from the URL encoded variables and populate the topic map by inserting the information into the appropriate database tables.

If instead the user chooses to add parts of a document to the trail, additional actions have to be taken. Firstly as a prerequisite for it to function, when opening a document and displaying it into one of the document windows, they are first parsed by PHP to add indexes to the elements found in the document. Also, in case of HTML documents, to follow Nelson's idea of stripping the document of any embedded markup, PHP strips the document of most markup elements, except for the <img> and <a> elements. This also has the side effect of clearing out any elements that might cause any ambiguities in indexing the content of the document, since the indexing is done by adding markup. The indexing itself is done by running through each piece of content in the document, marking them up with <**span**> elements assigning each of them a unique index number by using the *id*

attribute of the elements. At the same time event handlers are inserted, which will fire on mouse events, causing calls to a JavaScript function passing the elements id and content as an argument to it. These event handlers were meant to be triggered upon the selection of text by dragging the mouse to mark text. This could have been carried out by the native event handler in JavaScript called "onSelect" but it is only implemented to work for selection of text in form <input> elements. Rendering all text as form elements would have been quite inconvenient, and thus the solution for this prototype is to add an "onClick" event handler for the element where one has to click every word. It is not an optimal solution, but it is what was possible, and it illustrates the intended function in a real browser, where the user should be allowed to select ranges of text by dragging her mouse.

This JavaScript function appends its parameters to in-memory associative arrays correspondent with the two document windows, using the id of the element as the identifying key and assigning the word as its value. These arrays are then consulted when the user clicks the button labeled "Add selections to the trail" indicating that she wants to add whatever she selected in the document to the trail. The user is presented with dialogues allowing her to enter names and annotations for the nodes as well as associating the nodes to other nodes if there are any already existing nodes in the trail. All this information together with the selected pieces of information from the document itself is then sent to PHP which enters the information as topics and associations into the database.

10.4.4 Rendering the trail and its functions

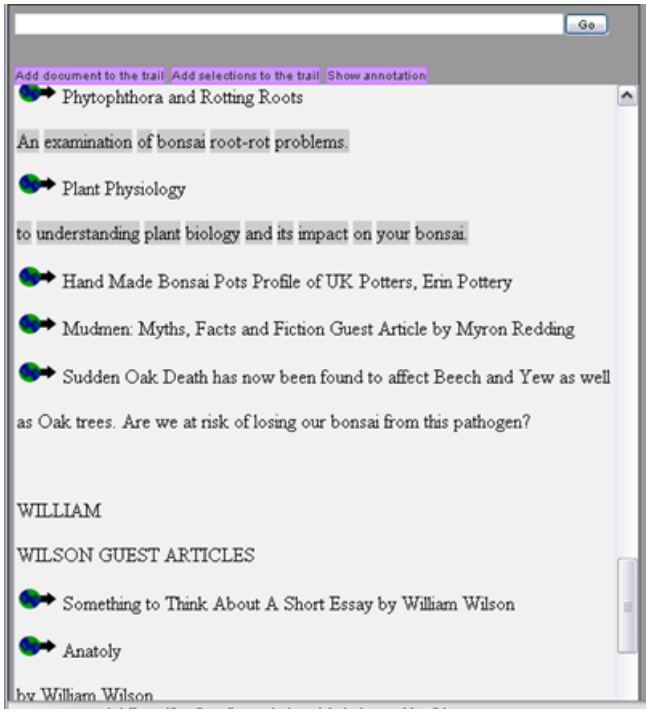
As soon as a trail is created and nodes are created and associated within it, it is ready for use. In fig. 10.3, the user first opens an existing trail, this is carried out as described above for the first use case, and the Touch Graph applet is rendered and displayed showing whatever nodes and associations exist in the trail. What could be thought of as a limitation of TouchGraph, is that it only renders nodes which play a role in some association. This means that there is no way of visualizing single nodes by the use of Touch Graph. Looking at Bushs's notion of the trail, nodes were added sequentially, and thus no node could exist in the trail before getting added as a part of an association. In this implementation, single nodes may be created without associating them to anything, they exist in the system and they can be associated to later if one wishes. Rendering them as single nodes hovering around the trail would probably not make any sense, and thus Touch Graph's approach for visualization is fitting for this application.

As the URI of any resource is added to the Topic Map trail as an occurrence of the topic, it is exported as the reference for the node when creating the XML read by the applet. Then, when the user clicks a node with her mouse, this URI is sent to a JavaScript function which again calls the function used to open URIs in the document windows described above. All annotations as well as selected text retrieved from the resource are represented as Topic Map occurrences as well, and are retrieved together with the rest of the information belonging to the node. In the case of annotations, place holders existing above the function buttons located above the document windows are populated with the information retrieved. To not overcrowd the interface the author has chosen to leave them invisible, but adding a new function button to the interface enabling the user to toggle the visibility of the annotations. In the case that the node represents some section of the document identified by the URI occurrence, the occurrence holding the selections and their corresponding id's is retrieved and parsed. The elements having id's corresponding to the id's found in the occurrence are dynamically highlighted by setting their Style Sheet "background" property to a color that makes the selected words stand out form the rest. Also, the offset coordinate of the first of the referenced selections is retrieved and used to scroll the document window to the position of the selection as seen in fig. 10.6

For this prototype, the author is not using the words found in the selections to check whether the words being highlighted actually are the same, something which would be necessary in a live browser to solve the problem of the changing nature of web documents discussed above, but out of the scope of this implementation. One could also imagine that it was to be possible to add several selections from the same resource, and then to have all of the selections highlighted in the document. One would then have to figure out some way of indicating that there are multiple selections in the rendered documents, since one can only scroll the document to one position at a time.

When the user decides that she needs to export the trail, in order to send the trail to somebody else, the trail is exported into the XML Topic Map notation. This way the path could be imported into an other user's TMemex, as well as being used by other Topic Maps aware software. Importing an external trail into the TMemex is carried out by the Topic Map engine which parses the XTM file and then inserts into the database. Likewise, the merging of trails is carried out by the Topic Map engine, which extracts the trails from the database and then merges them making any topic with the same PSI merge into one. The PSI's are assigned on creation of the nodes, and consist of the URI's of the resources they represent, as their uniqueness secure the PSI's uniqueness.

Figure 10.6: Screen shot of highlighting of in-document selections



10.5 Future possibilities

The TMemex is a just a proposal like the Memex itself, but it shows the possibilities posed by having subject centered, high level information implementing a persons personal path through existing information. This way it is possible to overcome the problems caused by poorly structured and marked up information while waiting for better solutions to these low level tasks. Bush's proposition of the trail as a better structuring of information for the individual is as valid now as then, as it does not require for information in itself to be structured or altered in a certain way, but instead creates a high level abstraction to map the information to a desired structure. Topic Maps have shown themselves to be an adequate tool for implementing this high level view of information, as it easily implements all the wanted features of the thought trail, and poses powerful additional possibilities as well through scoping, merging and typing.

As I discussed in 9.2.2, one could imagine the role of Topic Maps on the World Wide Web to be a top-down approach to information on the web, for the design of information into knowledge structures suited for human consumption. It may then very well be imagined that by using Topic Maps, the role of the browser could change from a passive to a more active medium for the users. Giving the user the possibility to create her own trails on top of possibly Semantic Web-enriched content, accustoming information structures for her own needs. Also, with the power inherent in Topic Map merging, one could imagine browsers interacting with Topic Map content implemented on the web itself, or even with the TMemexes of other users, based on the scheme of Instant Messaging, where users could connect their browsers up against each other directly. This would indeed open a new dimension in sharing and collaboration, getting closer to what Vannevar Bush hoped for.

Chapter 11

Summary and conclusion

Information overload is not a recent problem that was born with the birth of the World Wide Web. It has been recognized for a long time, and most notably by Vannevar Bush more than 60 years ago, and presented in his famous 1945 article "As we may think". He thought that humanity would not be able to develop any further if we did not do something to structure our dispersed and ever growing mass of information in a way that would help us use it to our common good. Bush saw the indexing methods of his time as insufficient for the task at hand, and proposed the structuring of information associatively, closer to how the human mind actually works, by arbitrary association. In light of all the new technology being created as a result of the war effort he envisioned the possibilities to use these technologies in the aid of man to possibly implement a machine that would help us in selecting our information associatively rather than by indexing. Thus the notion of the Memex, and its essential feature of being able to tie items of information together associatively into what he called a trail. One could say that Bush's vision and hopes for the future has inspired several of the people we may now call hypertext pioneers since Bush's time up until our present time. Inspired them to further develop the idea of associative structuring of information culminating in our current World Wide Web and HTML.

Ideas of associative structuring of information has since then been envisioned in different forms. Ted Nelson rejected any sequential structuring of information as unnatural in favor of pure associative structuring, and coined the term "hypertext" in 1965 as "non-sequential" writing. Nelson has later expressed his annoyance with the imposition of hierarchical structures upon information, because it is creating a limit for what other structures can be expressed using the resources. But he saw early that associative structuring alone leads to new information overload related problems such as becoming "lost in hyperspace" due to loss of context and disorientation. And has himself proposed and envisioned several schemes for ad-

vanced structuring of hypertext. Douglas Engelbart and other hypertext pioneers have seen the same or similar problems, and have created hypertext systems implementing advanced hypertext functionality to combat these problems in several ways by combining structured and associative structures. They have also seen the need as well for adding rich descriptions for elements such as information nodes and associations to aid the user of hypertexts to be able to select information more effectively as Bush envisioned, as well as preventing disorientation and overload.

The advanced hypertext functionalities identified in the field of hypertext research were sacrificed to ensure generality and the broad adaptation of the early World Wide Web when Tim Berners-Lee invented and created it in the beginning of the 1990's. And since then it has become increasingly apparent that HTML, the standard markup language for the web is, by implementing pure associative structuring and linking alone, not an adequate tool for increasing selectability or decreasing information overload. This is partly due to its inadequate descriptive power in assigning metadata to the content of resources, and partly due to its incorrect handling by both document authors and browser implementations alike. That this solution was ill adapted for our future informational needs was recognized at some point by the World Wide Web Consortium, and measures were taken in the form of the creation of new standards and frameworks, such as CSS, XML, RDF, and the vision of the Semantic Web. All of them thought to do their part in fulfilling the prerequisites for decreasing information overload and giving us tools for proper structuring of information. In addition, many others within the field of knowledge management and information design are contributing by creating new approaches and technologies to provide a basis for better solutions to information overload in the future.

We are in some ways in the same situation as Bush was in when he saw the problems that had to be solved, and at the same time seeing possible solutions being within our grasp by the use of new emerging technologies. Our task is now to figure out how to put these new technologies of our time at work to do what they are best at, and what roles they should play in the future framework for structuring of information on the web. We are though in a more favorable situation than Bush in that we have the advantage of hindsight, and can learn a lesson from seeing that new technology alone is not enough to solve our problems. There is also the need for a methodological approach to information structuring in order to consciously design holistic information that will work to the benefit of humanity.

This thesis is an attempt to look back at what we have learned about hypertext since Bush's time, and a discussion of possible solutions that could be attempted

by using new technologies to implement what has been seen to work well. As a practical exercise, an implementation of Bush's thought Memex and especially the notion of his trail is attempted by the use of Topic Map technology. This is done to illustrate how the need for a high level structuring of existing information was recognized already by Bush in envisioning the trail. And how Topic Maps could play an important role in implementing the high level information layer for a future web of information.

In conclusion, I will say that our future solution for our problem of information overload depends on what we manage to learn from our past errors. Vannevar Bush wanted the essence of information to come through by structuring information resources closer to how the human mind works, and he saw one essential dimension of it, namely associative thinking. What has been lacking on the web and in our information culture generally, is the structuring of information on other dimensions, and the high level information to bind it all together to convey the real meaning and essence of our information resources. I hope that the TMemex is a step on the way to show how we could have the best elements from old approaches merge with new approaches to create possible solutions for current and future needs. As our needs will surely change in the future, so will approaches change along the way, and we all have to cooperate and consciously design information to meet the goal that Bush and several of the hypertext pioneers recognized as most important, namely augmenting the human intellect through shared knowledge and collaboration.

Appendix A

SQL schema for the Topic Map Trail

```
# Host: localhost
# Database: topicmap
# Table: 'association'
#
CREATE TABLE `association` (
  `id` int(11) NOT NULL auto_increment,
  `topicmapid` int(11) NOT NULL default '0',
  `assoctype` varchar(100) NOT NULL default '',
  `scopes` varchar(100) default '',
  `members` varchar(100) NOT NULL default '0',
  PRIMARY KEY  (`id`),
  UNIQUE KEY `Unique` (`assoctype`)
) TYPE=MyISAM;

# Host: localhost
# Database: topicmap
# Table: 'basename'
#
CREATE TABLE `basename` (
  `id` int(11) NOT NULL auto_increment,
  `scopeid` varchar(100) default NULL,
  `baseName` varchar(200) NOT NULL default '',
  `sortname` varchar(100) default '',
  `displayname` varchar(100) default '',
  `topicId` varchar(100) NOT NULL default '0',
  PRIMARY KEY  (`id`)
) TYPE=MyISAM;

# Host: localhost
# Database: topicmap
# Table: 'member'
#
CREATE TABLE `member` (
  `id` int(11) NOT NULL auto_increment,
  `roletype` varchar(100) NOT NULL default '',
  `topicref` varchar(100) NOT NULL default '',
  PRIMARY KEY  (`id`)
) TYPE=MyISAM;

# Host: localhost
# Database: topicmap
# Table: 'occurrence'
```

```
#
CREATE TABLE `occurrence` (
  `id` int(11) NOT NULL auto_increment,
  `topicid` varchar(100) NOT NULL default '',
  `resourceRef` varchar(200) default NULL,
  `resourceData` text,
  `scopeId` varchar(100) default NULL,
  `roletype` varchar(100) default '',
  PRIMARY KEY  (`id`)
) TYPE=MyISAM;

# Host: localhost
# Database: topicmap
# Table: 'scopes'
#
CREATE TABLE `scopes` (
  `scopeid` int(11) NOT NULL auto_increment,
  `topicref` varchar(100) NOT NULL default '',
  PRIMARY KEY  (`scopeid`),
  UNIQUE KEY `topicref` (`topicref`)
) TYPE=MyISAM;

# Host: localhost
# Database: topicmap
# Table: 'subjind'
#
CREATE TABLE `subjind` (
  `id` int(11) NOT NULL auto_increment,
  `subjectIndicator` varchar(100) NOT NULL default '',
  PRIMARY KEY  (`id`)
) TYPE=MyISAM;

# Host: localhost
# Database: topicmap
# Table: 'subjref'
#
CREATE TABLE `subjref` (
  `id` int(11) NOT NULL auto_increment,
  `subjectResourceRef` varchar(100) NOT NULL default '',
  PRIMARY KEY  (`id`)
) TYPE=MyISAM;

# Host: localhost
# Database: topicmap
# Table: 'subjtopref'
#
CREATE TABLE `subjtopref` (
  `id` int(11) NOT NULL auto_increment,
  `subjectTopicRef` varchar(100) NOT NULL default '',
  PRIMARY KEY  (`id`)
) TYPE=MyISAM;

# Host: localhost
# Database: topicmap
# Table: 'topic'
#
CREATE TABLE `topic` (
  `id` int(11) NOT NULL auto_increment,
  `topicmapid` int(11) NOT NULL default '0',
  `topicid` varchar(100) NOT NULL default '',
  `psi` varchar(100) default '',
  `basenameid` varchar(100) NOT NULL default '',
```

```
  `subjectIndicatorid` varchar(200) default NULL,
  `subjectResourceid` varchar(100) default '',
  `subjectTopicRefid` varchar(100) default '',
  `occurrences` varchar(100) default '',
  `typeId` varchar(100) default NULL,
  PRIMARY KEY  (`id`),
  UNIQUE KEY `unitopid` (`topicid`)
) TYPE=MyISAM;

# Host: localhost
# Database: topicmap
# Table: 'topicmaps'
#
CREATE TABLE `topicmaps` (
  `id` int(11) NOT NULL auto_increment,
  `name` varchar(100) NOT NULL default '',
  `description` varchar(100) NOT NULL default '',
  `date` date NOT NULL default '0000-00-00',
  `version` varchar(100) NOT NULL default '',
  `author` varchar(100) NOT NULL default '',
  PRIMARY KEY  (`id`),
  UNIQUE KEY `unik_name` (`name`)
) TYPE=MyISAM;

# Host: localhost
# Database: topicmap
# Table: 'type'
#
CREATE TABLE `type` (
  `id` int(11) NOT NULL auto_increment,
  `typeid` varchar(100) NOT NULL default '',
  PRIMARY KEY  (`id`)
) TYPE=MyISAM;
```

Appendix B

TopicMap Object Interface

```
<?php
class Topic {

  var $props;


  function Topic($nyid,$nytype,$psi,$bn,$subjIndRefs,$subjResRefs,$subjTopicRef,$occ){
    $this->props['id'] = $nyid;
    $this->props['type'] = $nytype;
    $this->props['baseNames'] = $bn;
    $this->props['subjIndicator'] = $subjIndRefs;
    $this->props['subjResource'] = $subjResRefs;
    $this->props['subjTopicRef'] = $subjTopicRef;
    $this->props['occurrences'] = $occ;
    $this->props['psi'] = $psi;
  }


  function getID(){
    return $this->props['id'];
  }

  function setID($nyid){
    $this->props['id'] = $nyid;
  }

  function getType(){
    return $this->props['type'];
  }

  function setType($nytype){
    $this->props['type'] = $nytype;
  }

  function getBaseNames(){
    return $this->props['baseNames'];
  }

  function setBaseNames($nyBaseNames){
    $this->props['baseNames'] = $nyBaseNames;
  }
```

```
   function getSubjectIndicatorRefs(){
    return $this->props['subjIndicator'];
  }

  function setSubjectIndicatorRefs($subjinds){
    $this->props['subjIndicator'] = $subjinds;
  }

  function getSubjectResourceRefs(){
    return $this->props['subjResource'];
  }

  function setSubjectResourceRefs($subjrefs){
    $this->props['subjResource'] = $subjrefs;
  }


  function getSubjectTopicRefs(){
    return $this->props['subjTopicRef'];
  }

  function setSubjectTopicRefs($topicrefs){
    $this->props['subjTopicRef'] = $topicrefs;
  }

   function getPSI(){
    return $this->props['psi'];
  }

  function setPSI($psis){
    $this->props['subjTopicRef'] = $psis;
  }

  function getOccurrences(){
    return $this->props['occurrences'];
  }

  function setOccurrences($nyOccurrences){
    $this->props['occurrences'] = $nyOccurrences;
  }


}

class BaseName{
  var $props;

  function chopScope($str){
    $slashpos = strpos($str,"/");
    $ut = "";
     if($slashpos > -1){
$baseArr = explode("/",$str);
$ut = $baseArr[0];
     }else{
       $ut = $str;
     }
     return $ut;
  }//end chopScope


  function baseName($sc,$basenames,$sortname,$dispname){
    $this->props['scope'] = $sc;
```

```
    $this->props['baseNameStrings'] = $basenames;
    $this->props['sortNameString'] = $this->chopScope($sortname);
    $this->props['dispNameString'] = $this->chopScope($dispname);

    //print "<br/>Stringen som kom gjennom: ".$str;
  }

  function getScope(){
    return $this->props['scope'];
  }

  function setScope($str){
    $this->props['scope'] = $str;
  }

   function getBaseNameStrings(){
    return $this->props['baseNameStrings'];
  }

  function setBaseNameString($str){
    $this->props['baseNameStrings'] = $str;
  }

 function getSortNameString(){
    return $this->props['sortNameString'];
  }

  function setSortNameString($str){
    $this->props['sortNameString'] = $str;
  }

 function getDisplayNameString(){
    return $this->props['dispNameString'];
  }

  function setDisplayNameString($str){
    $this->props['dispNameString'] = $str;
  }

}//end class BaseName

class Occurrence{
  var $props;

  function Occurrence($topicid,$roleType,$resourceRef,$resourceData,$sc){
    $this->props['topicid'] = $topicid;
    $this->props['roleType'] = $roleType;
    $this->props['resourceRef'] = $resourceRef;
    $this->props['resourceData'] = $resourceData;
    $this->props['scope'] = $sc;
  }

  function getTopicID(){
    return $this->props['topicid'];
  }

  function setTopicID($nyid){
    $this->props['topicid'] = $nyid;
  }

 function getRoleType(){
    return $this->props['roleType'];
```

```
  }

  function setRoleType($nyrt){
    $this->props['roleType'] = $nyrt;
  }


 function getResourceRef(){
   return $this->props['resourceRef'];
 }

 function setResourceRef($nyrf){
   $this->props['resourceRef'] = $nyrf;
 }


 function getResourceData(){
   return $this->props['resourceData'];
 }

 function setResourceData($nyrd){
    $this->props['resourceData'] = $nyrd;
 }

 function getScope(){
   return $this->props['scope'];
 }

 function setScope($nysc){
    $this->props['scope'] = $nysc;
 }
}//end class Occurrence

//----------------- ASSOCIATIONS

//Members:

class Member{

  var $props;

  function Member($topicref,$roletype){

    $this->props['topicref'] = $topicref;
    $this->props['roletype'] = $roletype;
  }

  function getTopicRef(){
    return $this->props['topicref'];
  }

  function setTopicRef($nytopref){
    $this->props['topicref'] = $nytopref;
  }

  function getRoleType(){
    return $this->props['roletype'];
  }

  function setRoleType($newroletype){
    $this->props['roletype'] = $newroletype;
  }
```

```
}//end class Member


class Association{

  var $props;

  function Association($predicate,$members,$scopes){
    $this->props['assoctype'] = $predicate;
    $this->props['members'] = $members;
    $this->props['scopes'] = $scopes;
  }


  function getAssoctype(){
    return $this->props['assoctype'];
  }

  function setAssoctype($newpred){
    $this->props['assoctype'] = $newpred;
  }

  function getMembers(){
    return $this->props['members'];
  }

  function setMembers($newmembers){
    $this->props['members'] = $newmembers;
  }

  function getScopes(){
    return $this->props['scopes'];
  }

  function setScopes($newscopes){
    $this->props['scopes'] = $newscopes;
  }
}//end class Association
?>
```

Appendix C

Javascript / AJAX code

```
/*** GLOBAL VARS ***/
kontextarray1 = new Array();
kontextarray2 = new Array();
mapid = "";

/***
function to create a pointer to either a Microsoft.XMLHTTP object
or an XMLHttpRequest object depending on the navigator used
prerequisite for doing AJAX GET or POST calls
***/
function createRequestObject(){
return (window.ActiveXObject)? new ActiveXObject("Microsoft.XMLHTTP") : new XMLHttpRequest();
}//end function



/***
 function that opens the inserted url in the addressbar in one of the panes
     Makes an AJAX call to openURL.php which parses the URL, sets the activated
 panes innerHTML to be the result of the parsed content
 ARGS:
 @id => the id to the addressbar holding the url of the document to open
 @browser => the id of the textpane to open the document in
    ****/
function openURL(id,browser){
urlen = document.getElementById(id).value;
purl = "openURL.php?urlen="+urlen+"&browser="+browser;
var http = createRequestObject();
http.open('GET',purl,true);
http.onreadystatechange = function(){

if(http.readyState == 4 && http.status == 200){
var response = http.responseText;
if(browser == "browser1"){
if(document.getElementById("toggle1").innerHTML != ""){
document.getElementById("desc1").style.visibility = 'hidden';
document.getElementById("toggle1").innerHTML = "";
document.getElementById("desc1").style.display = 'none';
}//end if visbility
}else if(browser == "browser2"){
if(document.getElementById("toggle2").innerHTML != ""){
document.getElementById("desc2").style.visibility = 'hidden';
```

```
document.getElementById("toggle2").innerHTML = "";
document.getElementById("desc2").style.display = 'none';
}//end if visbility
}//end if browser
document.getElementById(browser).innerHTML = response;
}//end if
}//end function
http.send(null);
}//end function



/***
function that is called upon clicking a word in one of the panes
causing the id of the <span> element holding it to
be sent as an argument with the call
the function then picks up the word from the element and inserts into
the associative array "kontextarray" with its id as the key
ARGS:
@id => id of word
***/
function leggtilbit(id){
document.getElementById(id).style.background = "#cccccc";
idarr = id.split("_");
if(idarr[0] == "browser1"){
kontextarray1[id] = document.getElementById(id).innerHTML;
}else if(idarr[0] == "browser2"){
kontextarray2[id] = document.getElementById(id).innerHTML;
}//end if
}//end if


/***
function for creating a new trail
makes a call for send() with the argument of "show"
indicating that it should show the dialogue
***/
function startTrail(){
send('show','','','');
}//end function

/***
function doing the actual AJAX calls to "createMap.php" to create
new empty trails.
ARGS:
@arg => argument having the possibility of GET  or SHOW as values
@mapname => name of map / trail
@desc => description / annotation of trail
@auth => author of the trail
When @arg is SHOW, the function presents the user with a dialog where
she is asked to enter name, description of the trail as well as her name
When @arg is GET the trailarea is updated indicating that the trail was created
if the case was that the name the user entered already existed in the database,
the functions returns an error message saying so.
***/
function send(arg,mapname,desc,auth){
//create an XMLHttpRequestObject
var http = createRequestObject();

if(mapname != ""){
urlen = "createMap.php?arg="+arg+"&name="+mapname+"&desc="+desc+"&auth="+auth;
}else{
```

```
urlen = "createMap.php?arg="+arg;
}
http.open('GET',urlen,true);
http.onreadystatechange = function(){

if(http.readyState == 4 && http.status == 200){
var response = http.responseText;

if(response != ""){
if(arg == "show"){
document.getElementById("trail").innerHTML = response;
}else if(arg == "get"){
if(response.charAt(0) != "x"){
document.getElementById("tname").innerHTML = " : "+mapname+" [ "+desc+" ] Author: "+auth;
document.getElementById("trail").innerHTML = "The trail was created with successfully";
}else if(response.charAt(0) == "x"){
document.getElementById("trail").innerHTML = "<p>There was already a map with that name. <a href=\"javascrip
}//end if response x or not
}//end if showget
}//end if response at all
}//end if http.readyState
}//end function onreadystatechange
//send(null) because we are just ding GET requests.
http.send(null);
}//end function

/***
function cxtracting the information from the fialogue issued by send(show)
does a new call to send with "get" and the mapname, description and author
data enclosed as arguments.
***/
function doInit(){
mapname = document.trailform.name.value
desc = document.trailform.desc.value
auth = document.trailform.auth.value
send('get',mapname,desc,auth);
}



/***
function for closing a trail, makes a call for sendClose()
could as well have been done within this function,
but make it so to follow the pattern of the other functions
***/
function closeTrail(){
sendClose();
}//end function



/***
function doing the actual AJAX call to "closeMap.php",
to remove all sessionvariables pertaining to the currently
opened trail.
returns a message to the user that the trail has ben closed,
as well as resetting the trailarea by emptying the trailheader
and doing a page reload to have the browser acknowledge the
change in session state.
***/
function sendClose(){
//create an XMLHttpRequestObject
```

```
var http = createRequestObject();
urlen = "closeMap.php";
http.open('GET',urlen,true);
http.onreadystatechange = function(){

if(http.readyState == 4 && http.status == 200){
var response = http.responseText;

if(response != ""){
document.getElementById("trail").innerHTML = response;
document.getElementById("tname").innerHTML = "";
document.location = document.location;
}//end if response
}//end if http.readyState
}//end funciton onreadystatechange
//send(null) beacuse we ar doing a GET and not a POST
http.send(null);
}//end function


/***
function for exporting a trail, makes a call for sendExport()
could as well have been done within this function,
but make it so to follow the pattern of the other functions
ARGS:
@mid => the current mapid / trailid
***/
function exportTrail(mid){
sendExport(mid);
}//end func


/***
function doing the actual AJAX call to "exportGraphXML.php",
which parses the qurrent trail and exports into the XTM format
and serves it up for download
ARGS:
@mid => the current mapid / trailid // due to the nature of
***/
function sendExport(mid){
var http = createRequestObject();
urlen = "exportXTM.php?trail="+mid;
http.open('GET',urlen,true);
http.onreadystatechange = function(){

if(http.readyState == 4 && http.status == 200){
var response = http.responseText;

if(response != ""){
document.getElementById("trail").innerHTML = response;
//alert(response);
}
}
};
http.send(null);
}//end function


/***
function for opening a trail, makes a call for sendOpen()
with the argument of "show", indicating that it should
show the opening dialogue
```

```
***/
function openTrail(){
sendOpen('show','');
}//end func

/***
function doing the actual AJAX calls to "openTrail.php"
ARGS:
@arg => expecting "get" og "show" as values
@mapid => the id of the map extracted by doOpen
A GET request is done to openTrail.php which either produces
the openTrail dialogue or fetches the trail applet and
the name, description and author of the trail
depending on the "arg" url variable being set to "get" or "show"
***/
function sendOpen(arg,mapid){
//get a pointer to an XMLHttpRequest object
var http = createRequestObject();
var urlen = "openTrail.php?arg="+arg+"&mapid="+mapid;
//window.open(urlen);
http.open('GET',urlen,true);
http.onreadystatechange = function(){

if(http.readyState == 4 && http.status == 200){
var response = http.responseText;

if(response != ""){
if(arg == "show"){
document.getElementById("trail").innerHTML = response;
}else if(arg == "get"){
resarr = response.split("|");
document.getElementById("tname").innerHTML = " : "+resarr[0];
document.getElementById("trail").innerHTML = resarr[1];
document.getElementById("tdesc").innerHTML = "<span id=\"blank\">   [</span>"+resarr[2]+"<spa
document.getElementById("tauth").innerHTML = "<span id=\"blank\">  Author: </span>"+resarr[3];
document.getElementById("limenu").innerHTML = "<li><a href=\"javascript: \" onclick=\"closeTrail();return fa
document.location = document.location;
}//end if show or get
}//end if response
}//end if http.readyState
}//end function onreadystatechange
//send(null) because we are doing GET
http.send(null);
}//end function


/***
function for processing the users choice resulting from the
dialogue created with openTrail / sendOpen(show)
extracts the id of the chosen topicmap / trail,
removes the dialogue from the trailarea of the screen, and
makes a call for sendOpen with the argument of "get" and the
"mapid", indicating that it should fetch the trail and return it
***/
function doOpen(){
mapid = document.openform.map.options[document.openform.map.selectedIndex].value;
document.getElementById("trail").innerHTML = "";
sendOpen('get',mapid);
}//end fucntion
```

```
/***
function for adding whole documents to the trail
ARGS:
@id => id of the addressfield holding the url
of the document in question
A GET AJAX request is done to "getTrailname.php" which fetches the
current mapid / trailid from the PHP $_SESSION['mappid'] and returns it
to be sent as an argument together with the url to sendAddNode
with the argument "show" to indicate it chould show the
trail-making dialogue, if the mapid returned is null, then the
user is told that she has to select a trail to work on before trying
to add any details.
***/
function addDocument(id){
//create an XMLHttpRequestObject
var http = createRequestObject();
purl = document.getElementById(id).value;
urlen = "getTrailname.php";
http.open('GET',urlen,true);
http.onreadystatechange = function(){

if(http.readyState == 4 && http.status == 200){
var response = http.responseText;
mapid = response;
if(mapid > 0){
sendAddNode("show",mapid,purl,'','','','','','');
}else{
document.getElementById("trail").innerHTML = "<p>You must choose a trail before you can add documer
}//end if mapid
}//end if http.readyState
}//end function onreadystatechange
//send(null) beacuse we doing a GET and not a POST
http.send(null);
}//end function



/***
function for processing the users choice resulting from the
dialogue created with addDocument / sendAddNode(show)
extracts the id of the current topicmap / trail,
the url of the docuemnt being added to the trail, and the
name of the node given by the user, the description / annotation if there is one
removes the dialogue from the trailarea of the screen, and
makes a call for sendAddNode with the argument of "get" and the "mapid",
"url","description", and "nodename" indicating that it should
update the trail and return it
***/
function doAdd(){
mapid = document.nodeform.mid.value;
url = document.nodeform.url.value;
nodename = document.nodeform.nodename.value;
description = document.nodeform.description.value;
document.getElementById("trail").innerHTML = "";
sendAddNode('get',mapid,url,description,'','','','',nodename)
}//end function


/***
function for processing the users choice resulting from the
dialogue created with addDocument / sendAddNode(show) in the case
```

```
that there are more than one node in the trail, and thus, the user
has to create a link between the newl created node and one of the
existing nodes.
extracts the id of the current topicmap / trail,
the url of the document being added to the trail, and the
name of the node given by the user, the description / annotation if there is one
in addtion to doAdd, it also extracts the id of the node to associate to,
the names of the roles to be played in the association,
and a description of the association
removes the dialogue from the trailarea of the screen, and
makes a call for sendAddNode with the argument of "get" and the "mapid",
"url","description", and "nodename" ,"role1, "role2", "assocnode" and "assocdesc",
indicating that it should update the trail and return it
***/
function doAdd2(){
mapid = document.nodeform.mid.value;
url = document.nodeform.url.value;
nodename = document.nodeform.nodename.value;
description = document.nodeform.description.value;
assocnode = document.nodeform.assocnode.value;
assocdesc = document.nodeform.assocdescription.value;
role1 = document.nodeform.role1.value;
role2 = document.nodeform.role2.value;
document.getElementById("trail").innerHTML = "";
sendAddNode('get',mapid,url,description,assocnode,role1,role2,assocdesc,nodename);
}

/***
function does the actual AJAX calls for the doAdd and doAdd2 functions
ARGS:
@arg => expects "get" or "show" as values
@mapiden => the id of the map / trail to add to
@url => the url of the originating document
@description => the description / annotation of the node
@assocnode => the id of the node to link to
@role1 => name of the role to be played by the currently created node
@role2 => name of the role to be played by the node to be linked to
@assocdesc => description / annotation of the association
@nodename => name of the node to be created
If arg is set to "show", the user is presented with a dialogue where she
is asked to enter the required information needed to create the new node,
otherwise, if arg is set to "get", the php script will enter the data into the
topicmap being the underlying datamodel for the trail.
***/
function sendAddNode(arg,mapiden,url,description,assocnode, role1,role2,assocdesc,nodename){
//create pointer to an XMLHttpRequest object
var http = createRequestObject();

if(assocnode != ""){
urlen = "addNode.php?arg="+arg+"&assocdescription="+assocdesc+"&dokurl="+url+"&description="+description+"&a
}else{
if(arg == "show"){
urlen = "addNode.php?arg="+arg+"&mapid="+mapiden+"&dokurl="+url+"&nodename="+nodename+"&description="+descri
}else if(arg == "get"){
urlen = "addNode.php?arg="+arg+"&trailid="+mapiden+"&dokurl="+url+"&nodename="+nodename+"&description="+desc
}//end if arg
}//end if assocnode


http.open('GET',urlen,true);
http.onreadystatechange = function(){
```

```
if(http.readyState == 4 && http.status == 200){
var response = http.responseText;

if(response != ""){
if(arg == "show"){
document.getElementById("trail").innerHTML = response;
}else if(arg == "get"){
document.getElementById("tname").innerHTML = " : "+response;
}//end if showorget

}//end if response
}//end if http.readyState
}//end function onreadystatechange
//send(null)  since we are just doing a GET request
http.send(null);
}//end function

/***
function for adding selections done within a document to the trail
ARGS:
browser => the textpane where to choise is done
the global arrays kontextarray 1 & 2 contain the selections done,
and the browserid is used to determmine which array to pick data from
Then a GET AJAX request is done to "getTrailname.php" which fetches the
current mapid / trailid from the PHP $_SESSION[’mappid’] and returns it
to be sent as an argument together with the url and the selects to
sendAddNode2 with the argument "show" to indicate it chould show the
trail-making dialogue, if the mapid returned is null, then the
user is told that she has to select a trail to work on before trying
to add any details.
***/
function addSelects(browser){
arr= new Array();
if(browser == "browser1"){
arr = kontextarray1;
url = document.getElementById("urlen").value;
}else if(browser == "browser2"){
arr = kontextarray2;
url = document.getElementById("urlto").value;
}//end if

var utstr = "";
for(i in arr){
utstr += i+"|"+arr[i]+"d’";
}//end for

//create an XMLHttpRequestObject
var http = createRequestObject();
urlen = "getTrailname.php";
http.open(’GET’,urlen,true);
http.onreadystatechange = function(){

if(http.readyState == 4 && http.status == 200){
var response = http.responseText;
mapid = response;
if(mapid > 0){
sendAddNode2("show",mapid,url,utstr,’’,’’,’’,’’,’’,’’);
}else{
document.getElementById("trail").innerHTML = "<p>You must choose a trail before you can add documen
}//end if mapid
}//end if http.readystate
}//end function on readystatechange
```

```
//send(null) because we ar doing a GET REQUEST
http.send(null);
}//end function


/***
function for processing the users choice resulting from the
dialogue created with addSelects / sendAddNode2(show)
extracts the id of the current topicmap / trail,
the url of the document being added to the trail, and the
name of the node given by the user, the description / annotation if there is one
removes the dialogue from the trailarea of the screen, and
makes a call for sendAddNode2 with the argument of "get" and the "mapid",
"url","description", and "nodename" indicating that it should
update the trail and return it
***/
function doAdd3(){
mapid = document.nodeform.mid.value;
url = document.nodeform.url.value;
nodename = document.nodeform.nodename.value;
description = document.nodeform.description.value;
document.getElementById("trail").innerHTML = "";
sendAddNode2('get',mapid,url,'',description,'','','','',nodename);
}//end function


/***
function for processing the users choice resulting from the
dialogue created with addSelects / sendAddNode2(show) in the case
that there are more than one node in the trail, and thus, the user
has to create a link between the newl created node and one of the
existing nodes.
extracts the id of the current topicmap / trail,
the url of the docuemnt being added to the trail, and the
name of the node given by the user, the description / annotation if there is one
in addtion to doAdd, it also extracts the id of the node to associate to,
the names of the roles to be played in the association,
and a description of the association
removes the dialogue from the trailarea of the screen, and
makes a call for sendAddNode with the argument of "get" and the "mapid",
"url","description", and "nodename" ,"role1, "role2", "assocnode" and "assocdesc",
indicating that it should update the trail and return it
***/
function doAdd4(){
mapid = document.nodeform.mid.value;
url = document.nodeform.url.value;
nodename = document.nodeform.nodename.value;
description = document.nodeform.description.value;
assocnode = document.nodeform.assocnode.value;
assocdesc = document.nodeform.assocdescription.value;
role1 = document.nodeform.role1.value;
role2 = document.nodeform.role2.value;
document.getElementById("trail").innerHTML = "";
sendAddNode2('get',mapid,url,'',description,assocnode,role1,role2,assocdesc,nodename);
}

/***
function does the actual AJAX calls for the doAdd and doAdd2 functions
ARGS:
@arg => expects "get" or "show" as values
@mapiden => the id of the map / trail to add to
@url => the url of the originating document
```

```
@streng => the concatenated selected words and ids
@description => the description / annotation of the node
@assocnode => the id of the node to link to
@role1 => name of the role to be played by the currently created node
@role2 => name of the role to be played by the node to be linked to
@assocdesc => description / annotation of the association
@nodename => name of the node to be created
If arg is set to "show", the user is presented with a dialogue where she
is asked to enter the required information needed to create the new node,
otherwise, if arg is set to "get", the php script will enter the data into the
topicmap being the underlying datamodel for the trail.
The function uses POST as method since the url encoded string might, and will
become too long for a GET request due to the streng argument.
***/
function sendAddNode2(arg,mapiden,url,streng,description,assocnode, role1,role2,assocdesc,nodename)
var http = createRequestObject();
var params;
if(assocnode != ""){
urlen = "addNode2.php";
params = "arg="+arg+"&ad="+assocdesc+"&dokurl="+url+"&desc="+description+"&an="+assocnode+"&trailic
}else{
if(arg == "show"){
urlen = "addNode2.php";
params = "arg="+arg+"&mapid="+mapiden+"&dokurl="+url+"&nodename="+nodename+"&description="+descript
}else if(arg == "get"){
urlen = "addNode2.php";
params = "arg="+arg+"&trailid="+mapiden+"&dokurl="+url+"&nodename="+nodename+"&description="+descri
}//end if showorget
}//end if assocnode

http.open('POST',urlen,true);
//needed to a POST request
http.setRequestHeader("Content-type","application/x-www-form-urlencoded");
http.setRequestHeader("Content-length",params.length);
http.setRequestHeader("Connection","close");
http.onreadystatechange = function(){

if(http.readyState == 4 && http.status == 200){
var response = http.responseText;

if(response != ""){
if(arg == "show"){
document.getElementById("trail").innerHTML = response;
}else if(arg == "get"){
document.getElementById("tname").innerHTML = " : "+response;
}//end if showorget

}//end if response
}//end if http.readyState
}//end function onreadystatechange
//send(params) beacuse we ar doing a POST request
http.send(params);
}//end function



/***
function for opening URL's when clicking a node in the trail
ARGS:
@url => the url to be opened
@topicid => the id of the topic representing the node
```

```
Relays control to chooseWindow, with the addtional argument
"show" to signal the chooseWindow to show a dialogue
***/
function openlink(url,topicid){
chooseWindow('show','',topicid,url);
}//end function


/***
function that does the actual AJAX call invoked by openlink
ARGS:
@arg => expects "show" or "get" as its value
@vindu => the pan to be used for display
@topicid => the id of the topic representing the node / resource to be opened
@url => the url of the resource
if given "get" as a value for arg, a dialoge is returned, asking the user
to choose the pane in which to display the resource
***/
function chooseWindow(arg,vindu,topicid,url){
var http = createRequestObject();
if(arg == "show"){
urlen = "chooseWindow.php?arg="+arg+"&topicid="+topicid+"&url="+url;
}else if(arg == "get"){
urlen = "chooseWindow.php?arg="+arg+"&vindu="+vindu+"&topicid="+topicid+"&url="+url;
}//end if showorget

http.open('GET',urlen,true);
http.onreadystatechange = function(){

if(http.readyState == 4 && http.status == 200){
var response = http.responseText;

if(response != ""){
if(arg == "show"){
document.getElementById("trail").innerHTML = response;
}//end if show
}//end if response
}//end if http.readyState
}//end function onreadystatechange
//http send(null) because we are doing a GET request
http.send(null);
}//end function

/***
function that harvests the information from the dialogue posed
by chooseWindow.
the harvested information is then sent to openURL.php which
parses the url and returns markedup code for display, which
is then displayed in the pane chosen by the user.
Then the trailarea is updated to show the trail applet again.
At the end, calls to getOccs() and getInternalOccs() is done to
update the descriptionfields, and / or process the in-docuement
information forund with a node.
***/
function doChoose(){
vin = document.form1.vindu.options[document.form1.vindu.selectedIndex].value;
topicid = document.form1.topicid.value;
url = document.form1.url.value;
gammel1 = document.getElementById("desc1").innerHTML;
gammel2 = document.getElementById("desc2").innerHTML;
if(vin == 1){
```

```
purl = "openURL.php?urlen="+url+"&browser=browser1";
}else if(vin == 2){
purl = "openURL.php?urlen="+url+"&browser=browser2";
}//end if vin

var http = createRequestObject();

http.open('GET',purl,true);
http.onreadystatechange = function(){

if(http.readyState == 4 && http.status == 200){
var response = http.responseText;
if(vin == 1){
document.getElementById("browser1").innerHTML = response;
}else if(vin == 2){
document.getElementById("browser2").innerHTML = response;
}//end if vin

}//end if http.readyState
}//end function onreadystatechange
http.send(null);

//update the trailarea with the trail-applet again
var http2 = createRequestObject();
http2.open('GET',"getApplet.php",true);
http2.onreadystatechange = function(){
if(http2.readyState == 4 && http2.status == 200){
var response2 = http2.responseText;
document.getElementById("trail").innerHTML = response2;
}//end if http.readyState
}//end function onreadystatechange
http2.send(null);

getOccs(topicid,vin);
getInternalOccs(topicid,vin);
}//ende function


/***
function that fetches occurrences for nodes fecthed with doChoose,
being annotations
ARGS;
@topicid => id of the topic representing the node
@vin => id of the pane in which the owning node is to be displayed
The function does an AJAX call to getOccs.php which will fetch the associated
occurrence and return them to the function which will populate the designated
annotation areas in the browser.
***/
function getOccs(topicid,vin){
var http = createRequestObject();
urlen = "getOccs.php?topicid="+topicid;
http.open('GET',urlen,true);
http.onreadystatechange = function(){

if(http.readyState == 4 && http.status == 200){
var response = http.responseText;
pipepos = response.indexOf("|");

if(vin == 1){
/*  if the annotation has a pipe found within its stringrepresentation
it has more than just annotation information in it */
if(pipepos == -1){
```

```
document.getElementById("desc1").innerHTML = response;
/*figure out wether the annotation divs are visible or not, if
  so hide it, because this is a new annotation*/
if(document.getElementById("desc1").style.visibility == 'visible'){
document.getElementById("desc1").style.visibility = 'hidden';
}//end if visibility

//update the interface with a button which toggles the annotation on or off
document.getElementById('toggle1').innerHTML = "<a href=\": \" onclick=\"showDesc('desc1','an1');return fals

}else if(pipepos > -1){
bits = response.split("#");
document.getElementById("desc1").innerHTML = bits[0];
/*figure out whether the annotation divs are visible or not, if
  so hide it, because this is a new annotation*/
if(document.getElementById("desc1").style.visibility == 'visible'){
document.getElementById("desc1").style.visibility = 'hidden';
}//end if visbility

document.getElementById('toggle1').innerHTML = "<a href=\"javascript: \" onclick=\"showDesc('desc1','an1');r
}//end if pipesymbol

}else if(vin == 2){
if(pipepos == -1){
document.getElementById("desc2").innerHTML = response;
if(document.getElementById("desc2").style.visibility == 'visible'){
document.getElementById("desc2").style.visibility = 'hidden';
}//end if visibility
document.getElementById('toggle2').innerHTML = "<a href=\"javascript: \" onclick=\"showDesc('desc2','an2');r
}else if(pipepos > -1){
bits2 = response.split("#");
document.getElementById("desc2").innerHTML = bits2[0];
if(document.getElementById("desc2").style.visibility == 'visible'){
document.getElementById("desc2").style.visibility = 'hidden';
}//end if visibility
document.getElementById('toggle2').innerHTML = "<a href=\"javascript: \" onclick=\"showDesc('desc2','an2');r
}//end ifn pipesymbol
}//end if vindow
}//end if http.readyState
}//end funciton onreadystatechange
http.send(null);
}//end function

/***
function that fetches in-document selections for nodes fecthed with doChoose
ARGS;
@topicid => id of the topic representing the node
@vin => id of the pane in which the owning node is to be displayed
The function does an AJAX call to getOccs.php which will fetch the associated
occurrence and return them to the function which will highlight the designated
words in the document, and then scroll the pane to the place within the document
as if having anchors.
***/
function getInternalOccs(topicid,vin){
var http = createRequestObject();
urlen = "getOccs.php?topicid="+topicid;
http.open('GET',urlen,true);
http.onreadystatechange = function(){

if(http.readyState == 4 && http.status == 200){
var response = http.responseText;
pipepos = response.indexOf("|");
```

```
if(vin == 1){
/*  if the annotation has a pipe found within its stringrepresentation
it has more than just annotation information in it */
if(pipepos > -1){
bits = response.split("#");
bitarr = bits[1].split("d'");
var anchortop = "";
for(v=0;v<bitarr.length;v++){
bit = bitarr[v];
vnextarr = bit.split("|");
tid = vnextarr[0];
idarr = tid.split("_");
id = "browser1_"+idarr[1]+"_"+idarr[2];
word = vnextarr[1];

if(v == 0){
anchortop = id;
}//end if anchor

if(id != "browser1_undefined_undefined"){
document.getElementById(id).style.background = "#cccccc";
}//end if the id does not exist in the document.
offsettop = document.getElementById(anchortop).offsetTop;
document.getElementById("browser1").scrollTop = (offsettop-100);
}//end for

}//end if pipesymbol

}else if(vin == 2){
/*  if the annotation has a pipe found within its stringrepresentation
it has more than just annotation information in it */
if(pipepos > -1){
bits = response.split("#");
bitarr = bits[1].split("d'");
var anchortop = "";
for(v=0;v<bitarr.length;v++){
bit = bitarr[v];
vnextarr = bit.split("|");
tid = vnextarr[0];
idarr = tid.split("_");
id = "browser2_"+idarr[1]+"_"+idarr[2];
word = vnextarr[1];
if(v == 0){
anchortop = id;
}//end if anchor
if(id != "browser2_undefined_undefined"){
document.getElementById(id).style.background = "#cccccc";
}//end if id doesn't exist
offsettop = document.getElementById(anchortop).offsetTop;
document.getElementById("browser2").scrollTop = (offsettop-100);
}//end for
}//end if pipesymbol
}//end if vindow
}//end if http.readyState
}//end function onreadystatechange
http.send(null);
}//end function


/***
Auxilliary functions for turning the annotatationarea's
visbility on or off.
```

```
***/
function showDesc(id,an){
document.getElementById(id).style.visibility = 'visible';
document.getElementById(id).style.display = 'inline';
document.getElementById(an).innerHTML = 'Hide annotation';
document.getElementById(an).setAttribute('onclick','hideDesc(\''+id+'\',\''+an+'\');return false');
}//function

function hideDesc(id,an){
document.getElementById(id).style.visibility = 'hidden';
document.getElementById(an).innerHTML = 'Show annotation';
document.getElementById(an).setAttribute('onclick','showDesc(\''+id+'\',\''+an+'\');return false');
}//function

/***
function for importing Trails
ARGS:
@mapid => The id of the curent map / trail

***/
function importTrail(mapid){
url = "importTrail.php?mid="+mapid;
window.open(url,"Import","width=800,height=600");
}//end function


/***
function for merging Trails

***/
function mergeTrail(){
sendMerge('show','','');
}//end function

/***
function that does the AJAX call for mergeTrail.php
ARGS:
@arg => expects a value of either "show" or "get"
@map1 => id of trail1
@map2 => id of trail2
***/
function sendMerge(arg,map1,map2){
var http = createRequestObject();
if(arg == "show"){
urlen = "mergeTrail.php?arg="+arg;
}else if(arg == "get"){
urlen = "mergeTrail.php?arg="+arg+"&map1="+map1+"&map2="+map2;
}//end if showorget

http.open('GET',urlen,true);
http.onreadystatechange = function(){

if(http.readyState == 4 && http.status == 200){
var response = http.responseText;

if(response != ""){
if(arg == "show"){
document.getElementById("trail").innerHTML = response;
}//end if show
}//end if response
}//end if http.readyState
}//end function onreadystatechange
```

```
//http send(null) because we are doing a GET request
http.send(null);
}
```

Appendix D

HTML implementation of the browser interface

```
<?php
/* set the cache expire to 30 minutes */
session_cache_expire(30);
$cache_expire = session_cache_expire();
session_start();
include "func.php";
print "<"."?"."xml version=\"1.0\" encoding=\"iso-8859-1\""."?".">";
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<link rel="Stylesheet" type="text/css" href="style/stil.css" />
<script language="javascript" type="text/javascript" src="func.js"></script>
<title>TMemex</title>
</head>
<body onunload="opener.resizeTo(screen.availWidth,screen.availHeight)">
<div id="container">
<div id="header">
<?php
include "trail.php";
?>
</div>
<div id="one">
 <input type="text" name="urlen" id="urlen" class="ufield" /><input type="button" value="Go" class="sufield"
 <span id="desc1"> </span>
<div id="paneone">
<span id="pan1"><a href="javascript: " onclick="addDocument('urlen');return false;">Add document to the trai
<div id="browser1"></div>
</div>
</div>

<div id="two">
<input type="text" name="urlto" id="urlto" class="ufield" /><input type="button" value="Go" class="sufield"
<span id="desc2"> </span>
<div id="panetwo">
<span id="pan2"><a href="javascript: " onclick="addDocument('urlto');return false">Add document to the trail
<div id="browser2"></div>
</div>
</div>
```

```
</div>
</body>
</html>
```

Bibliography

Berners-Lee, T. and Fischetti, M. (2000). *Weaving the Web, The original design and ultimate destiny of the World Wide Web*. Harper Business, 1 edition.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American*, 284(5).

Berners-Lee, T. J. (1989). Information management: A proposal, in-house technical document, cern. `http://www.w3.org/History/1989/proposal.html`. Read: 13.04.2006.

Berners-Lee, T. J. (1998). Short history. `http://www.w3.org/People/Berners-Lee/ShortHistory`.

Bieber, M., Vitali, F., Ashman, H., Balasubramanian, V., and Kukkonen-Oinas, H. (1997). Fourth generation hypermedia: Some missing links for the world wide web. *International Journal of Human Computer Studies*, 47(1):31–65.

Borum, O.-J. (2003). Metadata: The semantic web i teori og praksis. Master's thesis, Universitetet i Bergen.

Bush, V. (1945). As we may think. *Athlantic Monthly*, 176(1):101–108.

Bush, V. (1991). Memex revisited. In *From Memex to Hypertext, Vannevar Bush and the Mind's Machine*, chapter 2, pages 197–216. Academic Press Inc.

Cailliau, R. and Ashman, H. (1999). Hypertext in the web a history. *ACM Comput. Surv.*, 31(4es):35.

Cianciarini, P., Folli, F., Rossi, D., and Vitali, F. (2002). Xlinkproxy: external linkbases with xlink. In *DocEng '02: Proceedings of the 2002 ACM symposium on Document engineering*, pages 57–65, New York, NY, USA. ACM Press.

Conklin, J. (1987). A survey of hypertext. In *Hypertext '87*. ACM.

Consortium, W. W. W. (2001). Modularization of xhtml. http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/introduction.html#s_intro_whatismod. Read: 01.11.2005.

Consortium, W. W. W. (2004a). Owl web ontology language guide. http://www.w3.org/TR/2004/REC-owl-guide-20040210/. Read 21.03.2006.

Consortium, W. W. W. (2004b). Owl web ontology language overview. http://www.w3.org/TR/2004/REC-owl-features-20040210/. Read 21.03.2006.

Decker, S., Melnik, S., van Harmelen, F., Fensel, D., Klein, M. C. A., Broekstra, J., Erdmann, M., and Horrocks, I. (2000). The semantic web: The roles of XML and RDF. *IEEE Internet Computing*, 4(5):63–74.

Engelbart, D. (1962). Augmenting human intellect : A conceptual framework. Technical Report AFOSR-3233, Stanford Research Institute.

Engelbart, D. C. (1995). Toward augmenting the human intellect and boosting our collective iq. *Commun. ACM*, 38(8):30–32.

Garshol, L. M. (2002). What are topic maps. http://www.xml.com/pub/a/2002/09/11/topicmaps.html?page=1. Read: 09.03.2006.

Garshol, L. M. (2003). Living with topic maps and rdf, topic maps, rdf, daml, oil, owl, tmcl. In *XML Europe*. IDEAlliance, http://www.idealliance.org.

Garshol, L. M. (2004). Metadata? thesauri? taxonimies? topic maps! *Journal of Information Science*, 30(4):378–391.

Guescini, R., Karabeg, D., and Nordeng, T. (2005). A case for polyscopic structuring of information. In Maicher, L. and Park, J., editors, *Charting the Topic Maps Research and Applications Landscape*, volume LNAI3873 of *Lecture notes in artificial intelligence*, pages 125–138. Springer-Verlag.

Gulli, A. and Signorini, A. (2005). The indexable web is more than 11.5 billion pages. In *14th International World Wide Web Confence*, pages 902–903.

Harold, R. E. and Means, S. W. (2002a). *XML in a nutshell*, chapter 2, page 24. O'Reilly & Associates, Inc.

Harold, R. E. and Means, S. W. (2002b). *XML in a nutshell*, chapter 6, page 89. O'Reilly & Associates, Inc.

help.Yahoo!.com (2006). How does the yahoo! directory differ from yahoo! search? http://help.yahoo.com/help/us/dir/basics/basics-03.html. Read: 10,2005.

Henzinger, M. (2005). Hyperlink analysis on the world wide web. In *HyperText '05*. ACM. Keynote.

Huynh, D., Mazzocchi, S., and Karger, D. R. (2005). Piggy bank: Experience the semantic web inside your web browser. In *International Semantic Web Conference*, pages 413–430.

Karabeg, D., Guescini, R., and Nordeng, T. (2005). Flexible and exploratory learning by polyscopic topic maps. In *5th IEEE International Conference on Advanced Learning Technologies*, pages 946–948. IEEE. http://www.win.tue.nl/SW-EL/2005/swel05-icalt05/final/W3-2.pdf.

Koivunen, M.-R. and Miller, E. (2002). W3c semantic web activity. In Hyvönen, E., editor, *Semantic Web Kick-Off in Finland, Vision, Technologies, Research and Applications*, pages 27–41. HIIT Publications.

Levy, D. M. (2005). To grow in wisdom, information overload and the life of leasure. In *JCDL '05*. ACM.

Manola, F. and Miller, E. (2004). *RDF Primer*. World Wide Web Consortium, http://www.w3.org/TR/2004/REC-rdf-primer-20040210, 20040210 edition.

Nelson, T. H. (1972). As we will think. In *From Memex to Hypertext, Vannevar Bush and the Mind's Machine*, chapter 3, pages 245–260. Academic Press Inc.

Nelson, T. H. (2005). Ted nelson's computer paradigm expressed as one-liners. http://xanadu.com.au/ted/TN/WRITINGS/TCOMPARADIGM/tedCompOneLiners.html. Read: 10.11.2005.

Nelson, T. S. (1990). *Literary Machines*, chapter 2, pages 43–45. Mindfull Press.

Nelson, T. S. (1993a). *Literary Machines*, chapter 1, page 18. Mindful Press.

Nelson, T. S. (1993b). *Literary Machines*, chapter 2, page 23. Mindful Press.

Nelson, T. S. (2001). Embedded markup considered harmful. This is a revised version of XML is evil.

Nielsen, J. (1999). Is navigation useful. http://www.useit.com/alertbox/991114.html. Read: 04, 2006.

Park, J. and Hunting, S., editors (2003). *XML Topic Maps, creating and using Topic Maps for the Web*, chapter 7, pages 124–127. Addison-Wesley, Pearson Education Inc.

Pepper, S. (2000). The tao of topic maps, finding the way in the age of infoglut. In *XML Europe, Paris France*. XML Europe, Paris France.

Pepper, S. (2004). Published subjects: Introduction and basic requirements. http://www.ontopia.net/tmp/pubsubj-gentle-intro.htm. Read: 26.04.2006.

Pepper, S. and Schwab, S. (2003). Curing the web's identity crisis. http://www.ontopia.net/topicmaps/materials/identitycrisis.hml. Read: 26.04.2006.

Powers, S. (2003). *Practical RDF*, chapter 1. O'Reilly Associates.

Practices, W. S. W. B. and Group, D. W. (2006). A survey of rdf/topic maps interoperability proposals. Technical report, World Wide Web Consortium.

Schraefel, M. C., Smith, D. A., Owens, A., Russell, A., Harris, C., and Wilson, M. (2005). The evolving mspace platform: leveraging the semantic web on the trail of the memex. In *HYPERTEXT '05: Proceedings of the sixteenth ACM conference on Hypertext and hypermedia*, pages 174–183, New York, NY, USA. ACM Press.

Simpson, R. (1996). 50 years after, as we may think : The brown/mit vannevar bush symposium. *Interactions...*

Toffler, A. (1970). *Future Shock*. Random House.

Trigg, R. H. (1991). From trailblazing to guided tours. In Nyce, J. M. and Kahn, P., editors, *From Memex to Hypertext, Vannevar Bush and the Mind's Machine*, chapter 3, pages 353–367. Academic Press Inc.

van Dam, A. (1988). Hypertext '87: keynote address. *Commun. ACM*, 31(7):887–895.

Wikipedia (2005a). Search engine optimization. http://en.wikipedia.org/wiki/Search_engine_optimization.

Wikipedia (2005b). Search engine spammer. http://en.wikipedia.org/wiki/Search_engine_spammer.

Wikipedia (2005c). Transclusion. "http://www.wikipedia.org/Transclusion". Read: 10.11.2005, last revision: 02.11.2005.

Wikipedia (2005d). Xml. http://en.wikipedia.org/wiki/Xml#History. Read: 10, 2005.

Wikipedia (2006a). Information overload. http://en.wikipedia.org/wiki/Information_overload. Read: 04, 2006.

Wikipedia (2006b). Knowledge integration. http://en.wikipedia.org/wiki/Kowledge_integration. Read: 27.04.2006.